SOFTAIR: SOFTWARE-DEFINED NETWORKING AND NETWORK FUNCTION VIRTUALIZATION SOLUTIONS FOR 5G CELLULAR SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Shih-Chun Lin

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2017

SOFTAIR: SOFTWARE-DEFINED NETWORKING AND NETWORK FUNCTION VIRTUALIZATION SOLUTIONS FOR 5G CELLULAR SYSTEMS

Approved by:

Dr. Ian F. Akyildiz, Advisor
Ken Byers Chair Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Raghupathy Sivakumar
Wayne J. Holman Chair Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Chuanyi Ji
Associate Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Pu Wang
Assistant Professor, Department of Electrical Engineering and Computer Science
Wichita State University

Dr. Geoffrey Ye Li
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: March 10, 2017

To my family,

for their endless love, support, and encouragement.

ACKNOWLEDGMENT

First of all, I would like to express my sincere gratitude to my advisor, Dr. Ian F. Akyildiz, for his invaluable guidance throughout my Ph.D. journey. He had been always sharing his immense knowledge and extensive experience without reservation. I am profoundly thankful to his limitless energy, passion, and encouragement, which not only guided me all the time of research, but also inspired me to embrace this career. I greatly appreciate Dr. Akyildiz to be my mentor. Without his persistent support, this achievement would have not been possible.

A very special gratitude goes out to all the professors and administrative staffs of the School of Electrical and Computer Engineering at the Georgia Institute of Technology.

With a special thanks to Dr. Chuanyi Ji and Dr. Geoffrey Ye Li, who provided constructive and valuable suggestions on my research work and kindly served in my Ph.D. Defense Reading Committee. Moreover, I am thankful to Dr. Raghupathy Sivakumar and Dr. Pu Wang (from Wichita State University), who kindly served in my Ph.D. Defense Committee. All of their insightful comments have helped me complete my Ph.D. degree and achieve a solid research path towards this thesis.

I am also grateful to all the former and current members of the Broadband Wireless Networking (BWN) Lab, for their consistent support and friendship. The family-like environment led me through the tough student life with great comfort. To all BWN Lab members, thank you for every moment that we shared together.

And finally, last but no means least, I can never find enough words to express my gratitude to my family, who have provided me through moral and emotional support in my life and supported me along the way. You are the champions!

SUMMARY

One of the main building blocks and major challenges for 5G cellular systems is the design of flexible network architectures which can be realized by the paradigm of software-defined networking (SDN) and network function virtualization (NFV). Existing commercial cellular systems rely on closed and inflexible hardware-based architectures both at the radio frontend and in the core network. These problems significantly delay the adoption and deployment of new standards, impose great challenges in implementing new techniques to maximize the network capacity and coverage, and prevent provisioning of truly-differentiated services for growing, uneven, and highly variable traffic patterns.

The objective of the thesis is to introduce an innovative software-defined architecture for 5G cellular systems, called SoftAir, via SDN and NFV solutions. The SDN concept has been proposed to efficiently create a centralized network abstraction with the programmability provisioning over the entire network. The complementary NFV concept has been also introduced to effectively virtualize functionality to run on cloud infrastructure by decoupling network functions and physical devices. Based on these two concepts, the SoftAir architecture enables the next-generation cellular networks with the needed flexibility for evolving and adapting to the ever-changing network context. In this thesis, first, a detailed overview is provided for priori wireless SDN (W-SDN) work. Second, the architecture design of SoftAir is introduced with key elements. Third, four essential management tools for SoftAir are developed, including control traffic balancing, optimal network planning, network virtualization, and traffic classifier. Fourth, the novel software-defined traffic engineering solutions enabled by SoftAir are presented, including base station clustering, throughput-optimal scheduling, and QoS-aware adaptive routing. Through the synergy of SDN and NFV solutions, the developed SoftAir in this thesis lays out the foundation for 5G wireless software-defined cellular systems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Existing commercial wireless networks are inherently hardware-based and rely on closed and inflexible architectural designs. Such inflexible hardware-based architectures typically lead to a 10-year cycle for a new generation of wireless networks to be standardized and deployed, impose significant challenges into adopting new wireless networking technologies to maximize the network capacity and coverage, and prevent the provision of truly-differentiated services able to adapt to increasingly growing, uneven, and highly variable traffic patterns. As shown in Figure 1, for 5G cellular system requirements, the ultra high capacity should have 1000-fold capacity/km^2 compared to LTE, the user-plane latency should be less than $1ms$ over the radio access network, and the ultra high data rates should provide 100-fold increase in user-experienced throughput (targeting $1Gbps$ experienced user throughput everywhere). The challenges faced by the current network architectures cannot be solved without a radical paradigm shift in the design of next-generation wireless networks. Hence, in this thesis, we propose the utilization of software-defined networking (SDN) and network function virtualization (NFV) concepts for next generation (5G) wireless networks, introduce a new architecture for wireless software-defined networks, called SoftAir, and present challenges and solutions for related research in this domain.

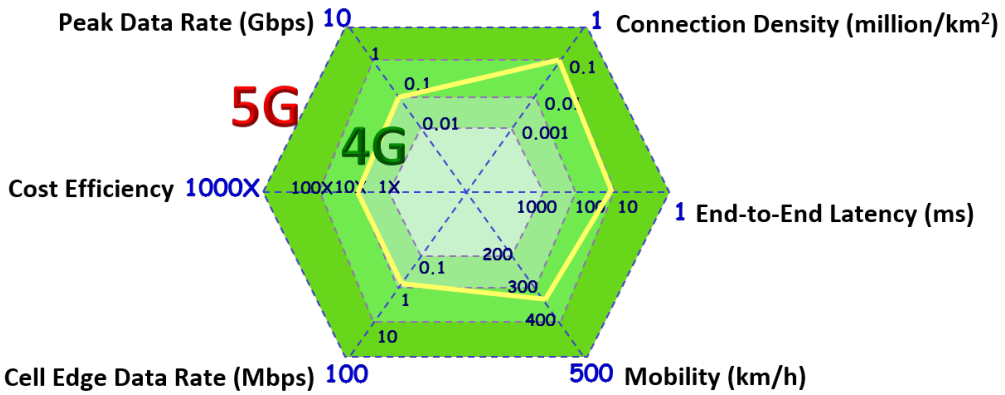


Figure 1. Cellular system requirements evolved from 4G to 5G.

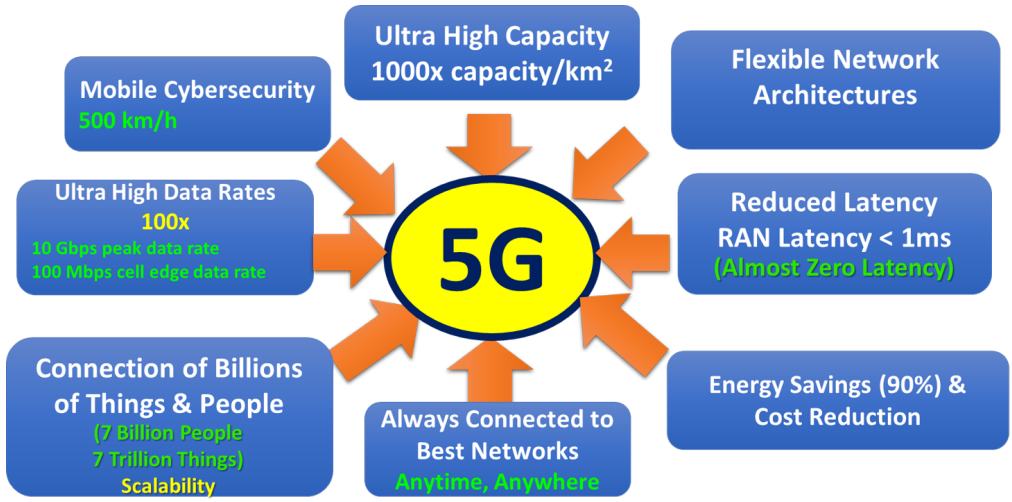


Figure 2. Design specifications for 5G cellular systems.

1.1 Requirements for 5G Cellular Systems

In conventional cellular networks, mobile phones were practically the only type of device expected to be supported. With the proliferation of Internet and its numerous applications, there was the problem of handling several classes of traffic to meet the different QoS requirements of diverse applications like video streaming, data, VoIP calls, etc. A similar situation is arising now with the need to support several types of devices and applications with drastically varying QoS requirement to provide better experience to the user. Unlike previous generations of cellular networks, 5G cellular network is envisioned to support a multitude of devices and applications like smartwatches, autonomous vehicles, Internet of Things (IoT), and tactile Internet. The various types of devices and application scenarios need more sophisticated networks that not only can support high throughput, but also provide low latency in data delivery, efficient energy consumption scheme, high scalability to accommodate a large number of devices, and ubiquitous connectivity for users. As illustrated in Figure 2, we describe these requirements in the following.

- **High Data Rates**: the metric of data rate has been the most important evaluation factor over generations of wireless communication networks. With the advent of mobile Internet and services such as HD video streaming, pervasive video and video sharing,

virtual reality available on mobile phones, as well as the proliferation of tablets and laptops which are accessible to wireless networks, increasing the data rate of cellular network is becoming an inevitable market driving force. Although the current maximum data rates can support HD video streaming which requires 8~15 Mbps, there are applications like ultra-HD 4K video streaming, high definition gaming, and 3D contents, which require even higher data rates at around 25 Mbps to provide a satisfactory experience to users. With these emerging applications demanding higher data rates, 5G networks are expected to have the peak data rate of around 10 Gbps which is a 100-fold improvement over current 4G networks [9]. Besides increasing the maximum data rate, the cell-edge data rate, as the worst case data rate users experience, should also be improved to 100 Mbps, which is a 100 times improvement over 4G networks at cell edge. The maximum data rate is an optimum estimate that a user can experience. In fact, the affects of intercell interference and transmission loss make the maximum value hardly achievable. Therefore edge data rate level becomes more important from the perspective of network engineering, as this data rate must support around 95% of users connected to the network. Another metric based on data rate that characterizes the network is the area capacity, which specifies the total data rate the network can serve per unit area. According to its definition, the unit of area capacity is normally bits per second per unit area. This metric is expected to increase 100 times in 5G compared to 4G network.

- **Low Latency**: the round-trip latency of data plane in the LTE network is around 15 milliseconds (ms) [10]. However, for the recently emerging applications such as tactile Internet, virtual reality, and multi-player gaming that 5G networks are expected to support, the latency should be upgraded to an order of magnitude faster than current network, at around 1 ms [11]. For instance, tactile Internet is a recently developed application where the wireless network is used for real-time control applications [12]. The latency required for such applications is determined by the typical

interaction for steering and control of real and virtual objects without creating cybersickness. The expected latency that would make these applications feasible is around 1 ms [12]. Although current smartphones have touch screens as the main interface, future devices will integrate various other interfaces like haptic, visual and auditory input and feedback, which will provide a new way of interacting with online environment for applications in virtual reality, healthcare, gaming, and sports, etc. These applications require real-time interactions with the user and any delay in the system will cause degradation to the user experience.

- **Low Energy Consumption**: the 5G networks are expected to support the IoT devices [13] which are basically some sensors that gather information about an environment and transmit it to a central server. These devices are mostly low-power, low-cost devices with lifespans as long as several years. Since these devices are not always connected to the base station and are only switched on occasionally, their battery life cannot afford the process of synchronization with the base station every time, as the synchronization step costs more energy than that of actual data transmission. This specific case in IoT requires that the radio access technique for 5G support loose or no synchronization. Moreover, this type of service also puts constraints on the computational power for decoding, the length of header, and packet forwarding scheme, etc. With the increasing number of connected smart devices, the number of base stations required to support these devices will also escalate. Because of the deployment of small cells, the base station will be densified. This foreseeable trend demands the base stations to be energy efficient since even a small improvement in energy efficiency will translate to huge energy savings in large scale.

- **High Scalability**: to support increasing amount of mobile devices that connect to the wireless network and communicate with each other, network scalability becomes an important factor in design of the next generation wireless communications network.

The increase in number of devices is further aggravated by the myriad of IoT devices and vehicle-to-vehicle communication technologies that are expected to surge in the 5G cellular network. Fueled by this smart equipment proliferation, it is expected that the number of devices connected to the cellular network will grow to 50 billion by 2020 [14]. Consequently, a highly scalable network that can efficiently accommodate this upsurge in number of devices is required. High scalability is also critical to performance of current and emerging applications, such as the IoT services, autonomous vehicles, etc. In the case of autonomous vehicles, prompt communications among them at high traffic densities necessitate the scalability of cellular network [15].

- **Improved Connectivity and Reliability**: apart from the aforementioned requirements, coverage and handover efficiency should also be improved for a better user experience, particularly when millimeter wave spectrum is exploited. With the increase in density of the base stations and the number of devices connected, as well as the introduction of femtocells and picocells, the number of handovers that the base station should handle will increase by at least two orders of magnitude. To support this demand, novel handover algorithms and techniques that provide improved coverage in cell edge areas are required. Another related issue is the authentication and privacy concerns related to the handover [16]. The delay to contact the authentication server for each handover will be hundreds of milliseconds which would be intolerable for 5G applications. Also, given the use of higher frequency bands in millimeter wave, the transmission range of signals is greatly reduced. Hence, maintaining connectivity becomes a great challenge for 5G. For mission-critical services, the requirements on high reliability and connectivity should always be guaranteed.

- **Improved Security**: the security aspect of wireless network recently attracts high attention, especially after 2015, when the applications of mobile payments and digital wallet became popular [17]. In retrospect of the previous generations of systems, the

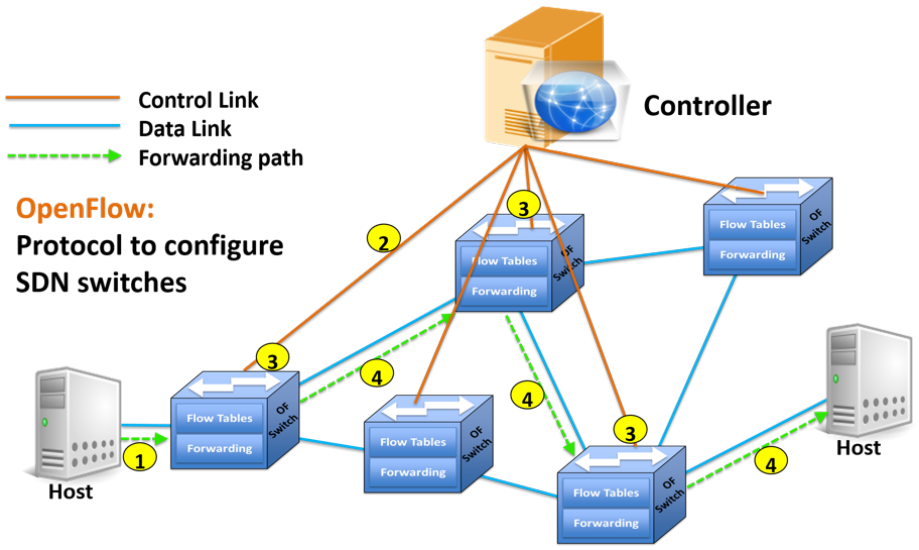


Figure 3. SDN operation.

general purpose of security is to protect basic connectivity and maintain user privacy. However, since the 5G system will ultimately face the dramatically increasing data traffic in the entire network, the requirement of security of 5G should not only be limited to providing trustworthy connectivity to users, but also improving the security on the whole network, addressing concerns on authentication, authorization as well as accounting, developing novel encryption protocols, and safeguarding cloud computing and management activities. For example, the security concerns are increasing since the introduction of near field communication (NFC) technique, which not only enables close proximity data transmission, but also may cause identity leaks. The 4G networks were not able to develop a unified standard to protect users' personal information, which will be fully addressed in the 5G networks.

1.2 Software-defined Networking and Network Function Virtualization

Software-defined networking (SDN) has been recently introduced primarily for data center networks and for the 5G Internet [18]. The main ideas are (i) to separate the data plane from the control plane and (ii) to introduce novel network control functionalities based on

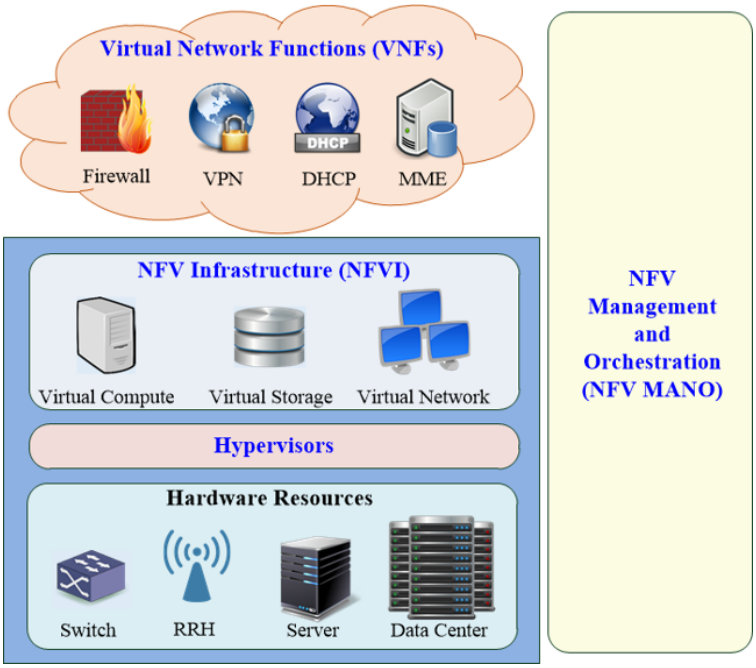


Figure 4. Overall architecture of NFV.

an abstract representation of the network. These are realized by (i) removing control decisions from the hardware (e.g., switches), (ii) enabling the hardware to be programmable through an open, standardized interface (e.g., OpenFlow [19]), and (iii) using a network controller to define the behavior and operation of network forwarding infrastructure. The SDN operation is shown in Figure 3. Specifically, when a flow arriving at switch does not match any rules in the flow table, it will be processed as follows: (1) the first packet of the flow is sent by the ingress switch to the controller, (2) the forwarding path for the flow is computed by the controller, (3) the controller sends the appropriate forwarding entries to install in the flow tables at each switch along the planned path, and (4) all subsequent packets in the flow or even different flows with matching (or similar) attributes are forwarded in the data plane along the path and do not need any control plane action. SDN makes it easier to introduce and deploy new applications and services; however, the effectiveness and great potential of SDN for 5G networking come with many new technical challenges, which need to be addressed by the new research advances [2].

Network function virtualization (NFV) enables the virtualization of entire network

functions (e.g. routing decisions) that were tied to hardware before to run on cloud infrastructure [20]. In conventional architectures, operators purchase and install proprietary devices to deploy each network function, while specialized hardware is usually very expensive but barely configurable. NFV, emerging as a breakthrough in 5G cellular systems, reduces CAPEX, OPEX, and power consumption through consolidating equipment and exploiting the economies of the IT industry scale [3]. According to ETSI [21], the overall architecture of NFV consists of four key elements: NFV infrastructure (NFVI), virtual network functions (VNFs), hypervisors, and NFV management and orchestration (NFV MANO), as shown in Figure 4. The main component here is VNFs which are software implementations of network functions, running on a generic cloud infrastructure. VNFs are deployed upon the NFVI that includes virtual computation, virtual storage, and virtual network resources. These virtual resources, created by hypervisors, bring virtualization over physical hardware resources within the network. Hardware resources might include networking (e.g., switches and RRHs), computing (e.g., server), and storage (e.g., data centers) infrastructures. The NFV MANO framework controls the provisioning of VNFs, the configuration of VNFs, and the infrastructure they run on. MANO can also chain several VNFs to activate an end-to-end service. NFV provides the availability of network appliance multi-version and multi-tenancy, allowing usages of a single platform for different applications, users, and tenants and enabling a wide variety of eco-systems with openness. However, the NFV development for wireless systems is still under-explored.

1.3 Research Objectives and Solutions

In this thesis, we propose the utilization of SDNs for next generation (5G) wireless networks and present a new architecture for wireless SDNs, called SoftAir, and the solutions and challenges for related research in this domain. In our proposed SoftAir architecture, the *control plane* consists of network management and optimization tools and is implemented on the network servers. The *data plane* consists of software-defined base stations

(SD-BSs) in the radio access network (RAN) and software-defined switches (SD-switches) in the cellular core network. Their control logic, e.g., physical/MAC/network functions, are implemented in software on general purpose computers and remote data centers.

Our proposed SoftAir architecture offers five core properties: (i) *programmability*, i.e., SDN nodes (e.g., SD-BSs and SD-switches) can be reprogrammed on-the-fly by dynamically associating with different network resources and networking algorithms; (ii) *cooperativeness*, i.e., SDN nodes can be implemented and aggregated at data centers for joint control and optimization to enhance the global network performance; (iii) *virtualizability*, i.e., multiple virtual wireless networks can be created on a single SoftAir, each of which operates under its own independent network protocols with network resources allocated based on demand; (iv) *openness*, i.e., data plane elements (i.e., BSs and switches), regardless of the underlying forwarding technologies and vendors, have unified data/control interfaces, e.g., CPRI and OpenFlow [19, 22], thus significantly simplifying the data plane monitoring and management; and (v) *visibility*, i.e., centralized controllers have a global view of the network status collected from BSs and switches. In the following, the challenges and developed solutions for SoftAir are briefly discussed.

1.3.1 Priori Art

There is an urgent need to study the fundamental architectural principles underlying a new generation of software-defined cellular network as well as the enabling technologies that supports and manages such emerging architecture. The first contribution of this thesis (Chapter 2) is an overview of the state-of-the-art W-SDNs solutions along with their associated NFV techniques. Specifically, we give the major problems of scalability challenges and vendor-specific device configuration, facing by the current cellular architectures. We also highlight the key differences among the existing W-SDN and NFV solutions, including SD-Wifi, programmable data plane, SD-MAC in WANs, SD-CN, SD-RAN, integrated SD-CN & SD-RAN, and their limitations.

1.3.2 Architecture Design

To the best of our knowledge, SoftAir is the first comprehensive solution suite for 5G cellular systems that accelerates the innovations for both hardware forwarding infrastructure and software algorithms, enables efficient and adaptive resource sharing, achieves maximum spectrum efficiency, encourages the convergence of heterogeneous networks, and enhances energy efficiency. The overall architecture of SoftAir for W-SDN in 5G systems composes of a data plane and a control plane. The data plane, which includes SD-RANs and a SD-CN, is an open, programmable, and virtualizable forwarding infrastructure. The control plane mainly consists of two components, which are network management tools and customized applications from service providers or virtual network operators.

The second contribution of this thesis (Chapter 3) is that we introduce SoftAir architecture and provide a completed qualitative comparison between SoftAir and existing architectures. In particular, we consider four key design elements of scalable SoftAir architecture as listed in the following. (1) Scalable software-defined planning that SoftAir decouples control and data planes for both SD-RANs and the SD-CN. (2) Fine-grained fronthaul network decomposition that SoftAir simultaneously realizes the physical-, MAC-, and network-layer function virtualization for RAN and adopts a new fine-grained fronthaul network decomposition architecture by leaving partial baseband processing at the RRH (e.g., modulation/demodulation), while implementing the remaining baseband functions (e.g., source coding and MAC) at the BBS. (3) Seamless OpenFlow incorporation that SoftAir incorporates OpenFlow protocol into SD-BSs and promises the transparent interconnections between SD-CNs and SD-RANs for the unified management over the entire network. (4) Network virtualization capacity that SoftAir enables the end-to-end network virtualization traversing both SD-RAN and SD-CN, realizing a truly multi-service converged network infrastructure. Moreover, we also provide a qualitative comparison of existing W-SDN solutions and SoftAir in terms of architecture, scalability, network virtualization, traffic engineering solutions, and research community.

1.3.3 Management Tools

Cloud orchestration aims to automate the configuration, coordination and management of software and software interactions in the cloud environment. To support cloud orchestration in SoftAir, to enable the promising features and to maximize the overall performance of SoftAir, several essential and general management tools need to be developed.

The third contribution of this thesis (Chapter 4) is that we propose four essential management tools to realize the promising properties of SoftAir, as listed in the following. (1) In-band control traffic balancing that finds the optimal control traffic forwarding paths for each SD-switch/SD-BS in such a way the average control traffic delay in the whole network is minimized. (2) Traffic-driven optimal network planning that jointly optimizes control traffic balancing and controller placement so that the required controllers and the control traffic delay are minimized. (3) Resource-efficient network virtualization that enables the jointly optimized design of QoS-aware virtualization and routing by tenant isolation and prioritization as well as flow allocation, fulfilling QoS requirements of tenants' applications. (4) QoS-aware traffic classifier that jointly exploits deep packet inspection and semi-supervised machine learning so that accurate traffic classification can be realized, while requiring minimal communication between the network controller and SD-switches (or SD-BSs).

1.3.4 Software-defined Traffic Engineering

Several new traffic engineering solutions are designed to leverage the full potential of the SoftAir architecture. Specifically, BS clustering, throughput-optimal scheduling, and QoS-aware routing solutions are proposed, which directly supported by the enabling tools discussed in Chapter 4.

The fourth contribution of this thesis (Chapter 5) is that we develop four novel software-defined traffic engineering solutions for SoftAir, as listed in the following. (1) Dynamic BS formation that treats the NLOS problem in millimeter-wave systems from SoftAir (a wireless software-defined networking architecture) perspective and adaptively coordinates BSs

and their multiple antennas to always satisfy UEs' QoS requirements in NLOS cases. (2) Delay-based maximum power-weight scheduling that brings throughput optimality (with respect to moment stability) for single-hop flows with heavy-tailed traffic. (3) Delay-based maximum-weight scheduling policy with the last-in first-out (LIFO) service discipline that lets a networked system can support the largest set of incoming traffic flows, while guaranteeing bounded queueing delay to each queue, no matter the queue has HT or LT traffic arrival. (4) QoS-aware adaptive routing as network service that supports diverse QoS requirements from user applications in packet delay, loss, and throughput in the same forwarding infrastructure.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. An overview of priori wireless software-defined networking (W-SDN) work is provided in Chapter 2. Then, the architecture design of SoftAir is introduced in Chapter 3. Moreover, the essential management tools for SoftAir are developed in Chapter 4. In addition, the software-defined traffic engineering solutions enabled by SoftAir are presented in Chapter 5. Finally, the research contributions are summarized in Chapter 6.

CHAPTER 2

STATE-OF-THE-ART WIRELESS SOFTWARE-DEFINED NETWORKING

In literature, the software-defined architectures are well-studied in wired networks. For example, in data center networks and campus local area networks (LANs) [19, 23, 24], these architectures mainly support centralized and adaptive manipulation of flow tables at switches and routers. Furthermore, considering wired network virtualization, cloud computing and computer virtualization have maintained strong foothold for the past few years. In particular, the virtualization of routers and switches has been adopted, such as virtual private networks (VPNs) over wide area networks (WANs) and metropolitan area network (MANs) as well as virtual LANs in enterprise networks. This is achieved by logically partitioning a physical network into virtual networks that share the physical routers/switches/crossconnects, physical links, and bandwidth on each link. The utilization of the physical resources needs to be carefully managed to maintain the QoS and security needs of the users of each virtual network. However, SDN's effectiveness and great potential for 5G data networking come with many new technical challenges, which need to be addressed by the new research advances.

2.1 Major Problems with Current Cellular Architectures

Figure 5 shows the current LTE network architecture and the corresponding data plane, which includes three components: cellular RAN, cellular CN, and the Internet. In particular, user equipment (UE) connects to eNodeB, i.e., base stations, and directs traffic through serving-gateway (S-GW) over a GPRS Tunneling Protocol (GTP) tunnel. S-GW serves as a local mobility anchor that enables seamless communication when the user moves from one BS to another. Towards this, S-GW must handle frequent changes in users' location, and store a large amount of user states since users retain their IP addresses when they move.

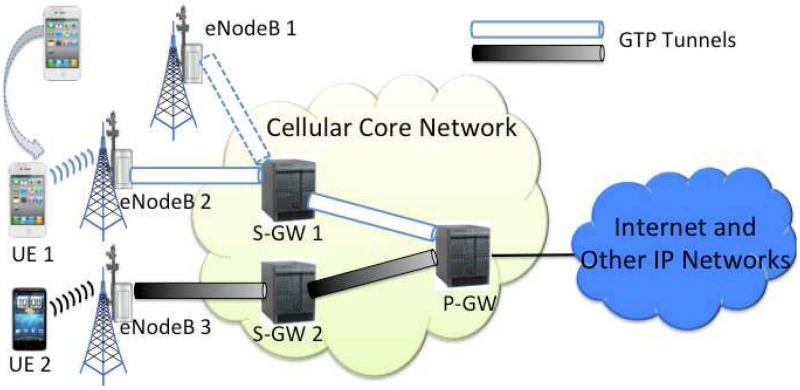


Figure 5. LTE network architecture and data plane. [1]

In addition, S-GW tunnels traffic to the packet-gateway (P-GW) which enforces QoS policies and monitors traffic to perform billing. P-GW also connects to the Internet and other cellular data networks, and acts as a firewall that blocks unwanted traffic. The bottomline here is that besides data plane functionalities, eNodeBs, S-GWs, P-GWs also participate in several control plane protocols. Specifically, with mobility management entity, these devices perform hop-by-hop signaling to handle session setup, tear-down & reconfiguration, and mobility, e.g., registration, paging, and handoff. Moreover, S-GW and P-GW are also involved in routing such as OSPF. The policy control and charging function (PCRF) manages flow-based charging in the P-GW. It also provides the QoS authorization that decides how to treat each traffic flow, based on the user profile. The home subscriber server (HSS) contains subscription information for each user In case of cell congestions, a BS in coordination with P-GW reduces the max rate.

The major problems with the current cellular architectures lie as follows: (1) **Scalability challenges** and (2) **Vendor-specific device configuration**. First, centralizing data-plane functions such as monitoring, access control, and QoS functionality at P-GW introduces scalability challenges. It cause the equipment becomes very expensive, e.g., more than $6M$ for a Cisco P-GW. Moreover, centralizing data-plane functions at the cellular-Internet boundary forces all traffic through the P-GW. It becomes difficult to host popular content inside the cellular network. Second, network equipment has vendor-specific configuration

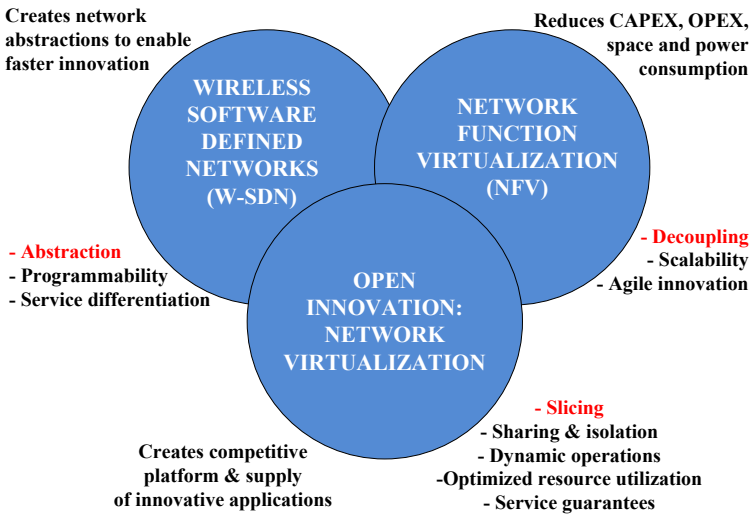


Figure 6. W-SDN relationship with NFV. An example of open innovation, e.g., network virtualization.

interfaces, and communicate through complex control-plane protocols, with a large and growing number of tunable parameters (alone several thousands parameters for base stations). Hence, carriers (operators) have (at best) indirect control over the operation of their networks, with little ability to create innovative services. In short, existing commercial cellular systems rely on closed and inflexible hardware-based architectures both at the radio frontend and in the CN. These problems significantly delay the adoption and deployment of new standards, impose significant challenges in implementing and innovation of new techniques to maximize the network capacity and accordingly the coverage, and prevent provisioning of truly-differentiated services which are able to adapt to growing and uneven and highly variable traffic patterns. To tackle these problems, in the following, we summarize the design of flexible network architectures for 5G cellular systems, which are realized by the SDN paradigm with NFV.

2.2 Trends to W-SDN and NFV

As 5G cellular systems will be driven by software, the new architecture solutions heavily rely on two emerging technologies, i.e., SDN and NFV as shown in Figure 6. In particular,

the benefit of SDN lies in its ability to provide an abstraction of the physical network infrastructure. Through network-wide programmability, the capability to change the behavior of the network as a whole, SDN greatly simplifies the management of networks. The level of network programmability provided by SDN allows several network slices to be customized and optimized for different service deployments, while using the same physical and logical network infrastructure. Furthermore, by separating network function from underlying hardware devices, NFV allows a network function to be implemented in software either locally or on the remote clouds. This capability can enhance network scalability, which allows the optimal organization and easy management of network control and monitoring tools over the whole network. The most significant benefit brought about by NFV is the flexibility to execute and improve network management functions timely and independently of the underlying physical network forwarding infrastructure.

Recently, few SDN architectures [1, 25–34] are exploited in wireless networks. OpenRoads [25] is mainly targeted at WiFi networks with little support for cellular networks. OpenRadio [26] proposes a novel programmable wireless data plane that provides modular programming capability for the entire wireless stack. However, OpenRadio does not provide any network controller that take advantage of such programmable data plane. CloudMAC [27] is a distributed network architecture that has a programmable MAC layer for 802.11 WLAN without resorting to software radios and partially performed MAC processing in data centers on virtual machines connected by an OpenFlow controlled network. Odin [28] is an SDN framework that proposes to simplify the implementation of high level enterprise WLAN services, such as authentication, authorization and accounting, by introducing light virtual access points. This approach is similar to the virtual access points used in CloudMAC. However, OpenRoads, OpenRadio, CloudMAC, and Odin are all partial solutions that exploit SDN concept in wireless networks and do not provide a completed/integrated architecture/platform for 5G cellular systems.

2.3 Integrated W-SDN & NFV Solutions

Integrated SDN solutions [1, 29–34] have also been introduced for practical 5G system implementations. CellSDN [1] aims to achieve a centralized control plane for cellular core networks. ADRENALINE [29] provides an industrial solution of software-defined core network (SD-CN) for 5G cellular systems with optical OFDM. However, these architectures neither consider the scalability issue of SD-CN nor the incorporation of CN with RAN. On the other hand, SoftRAN [30] attempts to restructure the control plane of RAN in a software-defined manner. An emerging distributed RANs, called Cloud-RAN [31], proposes software-defined radio access network (SD-RAN) architecture that connects SD-RANs to virtual BS pool through fibers and provides centralized control solution upon the BS pool. However, its coarse-grained BS decoupling limits the scalability and evolvability of such distributed RANs due to excessive, redundant I-Q transmissions. DOCOMO [32] and SK Telecom [33] also provide their own industrial SD-RAN solutions for 5G systems, respectively. CONTENT [34] is a 3-years European co-funded (FP7) project, which aims at offering a network architecture and overall infrastructure solution to facilitate the cross-technology virtualization in support of optimized, seamless and coordinated cloud, and mobile cloud service provisioning across heterogeneous network domains. However, similar to Cloud-RAN, these solutions (i.e., DOCOMO, SK Telecom, and CONTENT) adopt coarse-grained decoupling and thus bring bottlenecks to fronthaul fiber networks.

What is more important, all of the above architectures lack a coherent framework that fully integrates cellular CN and RAN in a software-defined manner. To this end, we propose a new W-SDN architecture that incorporates SDN and NFV and brings a coherent integration of SD-CN and SD-RANs for 5G cellular systems [2].

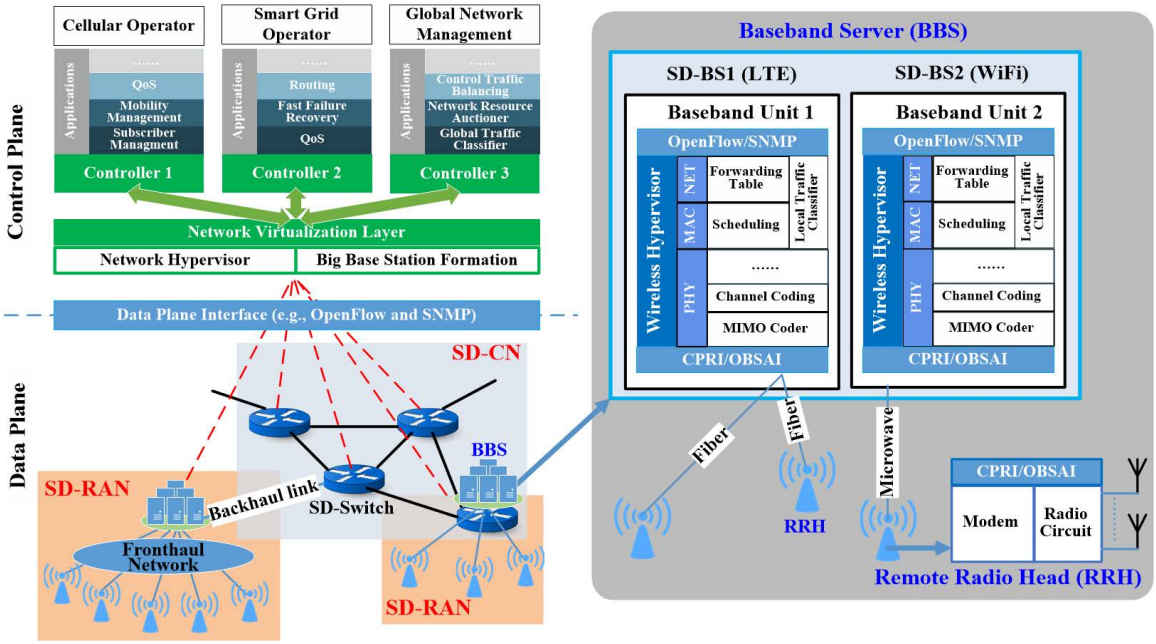


Figure 7. Network architecture of SoftAir [2].

CHAPTER 3

SOFTAIR ARCHITECTURE DEISGN

In this chapter, we introduce SoftAir architecture for 5G cellular systems. As shown in Figure 7, the architecture of SoftAir consists of a data plane and a control plane. The data plane is an open, programmable, and virtualizable network forwarding infrastructure, which consists of SD-RANs and a SD-CN. The SD-RAN consists of a set of SD-BSs, while the SD-CN is composed of a collection of SD-switches. The control plane mainly consists of two components: (i) network management tools and (ii) customized applications of service providers or virtual network operators. Four key elements of scalable SoftAir architecture are introduced.

3.1 Scalable Software-defined Planning

As shown in Figure 8, to increase network scalability, SoftAir decouples control and data planes for both SD-RANs and the SD-CN, and establishes an ubiquitous software-defined

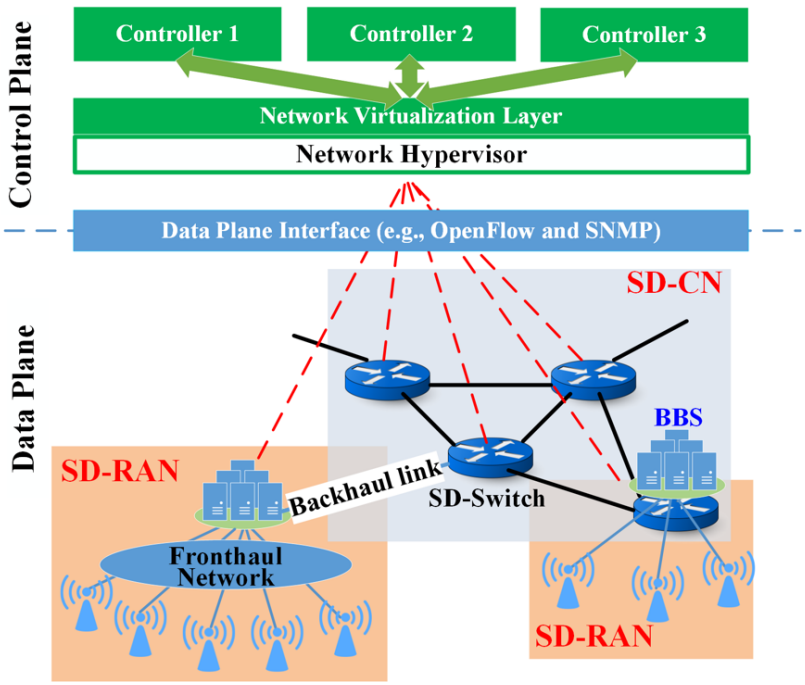


Figure 8. Scalable software-defined planning of SoftAir.

planning from RANs, the CN, to the Internet, effectively realizing multi-controller scenarios and optimum managements for large-scale wireless systems. In particular, SD-RAN consists of SD-BSs that jointly form a baseband server (BBS) and connect with numerous remote radio heads (RRHs). Moreover, by interconnecting physical links, the SD-CN contains SD-switches that have flow tables to route traffic. The control logic of SD-RANs (e.g., physical/MAC/network functions) and the SD-CN (e.g., network management and optimization tools) is implemented in software on general purpose computers, network servers, and remote data centers as high-performance controllers. Specifically, the network operating system in the control plane collects information about network states such as device status, link utilization, link delays, etc. The information collected is used to create a high-level network abstraction that helps the computation of optimum forwarding rules and efficient resource allocations. These control policies are then fed back to SD-BSs and SD-switches for execution through standard (south-bound) interfaces, e.g., OpenFlow and SNMP (Simple Network Management Protocol). Also, the control plane (or controllers) provides the network abstraction to applications, such as mobility management, security,

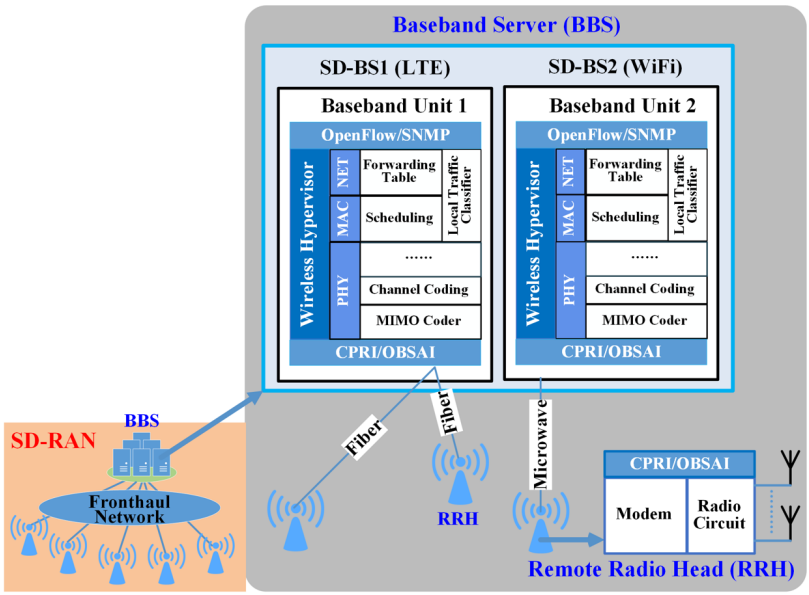


Figure 9. Fine-grained fronthaul network decomposition of SoftAir.

load balancing, etc., running upon it through (northbound) interfaces.

3.2 Fine-grained Fronthaul Network Decomposition

SoftAir supports fine-grained fronthaul network decomposition through dedicated base station NFV, and offers excellent cooperative gain and evolvability by allowing the aggregation of a large number of technology-evolving RRHs at BBS through the diverse, cost-efficient, CPRI (Common Public Radio Interface)-supported fronthaul network topologies, and over different fronthaul mediums. In particular, as in Figure 9, to enhance network flexibility, SoftAir simultaneously realizes the physical-, MAC-, and network-layer function virtualization for RAN and thus forms SD-RAN. Here, the proposed SD-RAN follows a distributed RAN architecture, and each SD-BS is split into hardware-only radio heads and software-implemented baseband units. These RRHs are connected to the baseband units on BBS through fronthaul network (fiber or microwave) using standardized interfaces, such as CPRI or OBSAI (Open Base Station Architecture Initiative) interface. What is more important, SoftAir adopts a new fine-grained fronthaul network decomposition architecture by leaving partial baseband processing at the RRH (e.g., modulation/demodulation), while

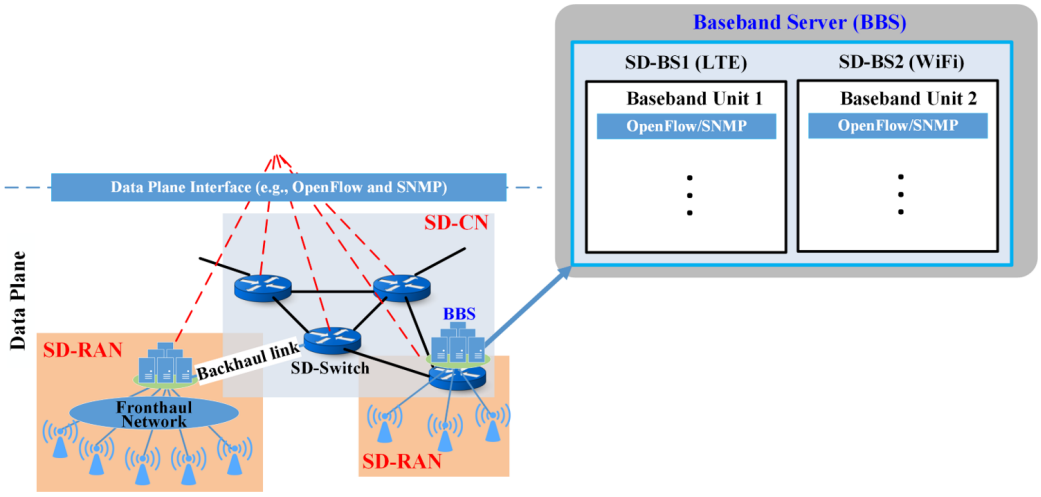


Figure 10. Seamless OpenFlow incorporation of SoftAir.

implementing the remaining baseband functions (e.g., source coding and MAC) at the BBS. Hence, the decomposition not only preserves sufficient flexibility offered by the distributed RAN architecture, but eliminates fronthaul network bottlenecks as well.

3.3 Seamless OpenFlow Incorporation

Different from the existing distributed RAN architecture, SoftAir incorporates the OpenFlow protocol into SD-BSs, and promises the transparent interconnections between SD-CNs and SD-RANs for the unified management over the entire SoftAir. Specifically, as in Figure 10, SD-RANs implement an OpenFlow interface for each SD-BS by utilizing Open vSwitch (OVS), which is an OpenFlow-capable software that can easily be realized in BBS. With OVS, each SD-BS can interpret, exchange, and respond to OpenFlow protocol messages. Equipping SD-BSs with OpenFlow capabilities provides an unified interface to control and manage base stations with different wireless standards, thus leading to a multi-technology converged RAN that allows smooth transitions among different radio access technologies.

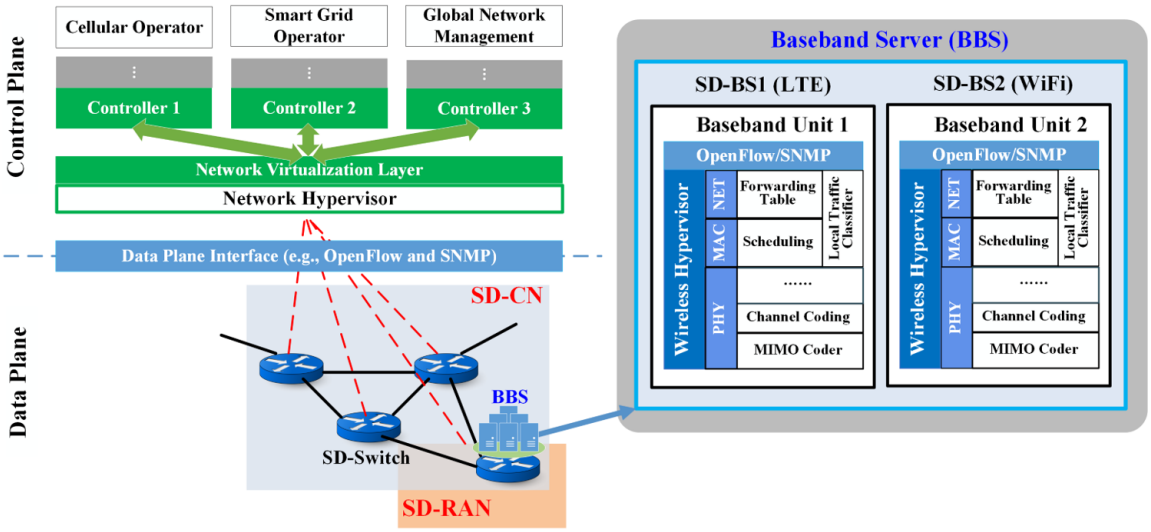


Figure 11. Network virtualization capacity of SoftAir.

3.4 Network Virtualization Capacity

SoftAir has the capability to enable the end-to-end network virtualization traversing both SD-RAN and SD-CN, thus realizing a truly multi-service converged network infrastructure. Specifically, as in Figure 11, the network virtualization enables multiple isolated virtual networks (e.g., M2M, smart-grid, over-the-top service provider, cellular provider) to share the same physical network infrastructure. That means it focuses on slicing network resources for multiple virtual networks so that they can simultaneously share the same physical network architecture. More specifically, each virtual network can adopt its customized physical-/MAC-/network-layer protocols, without interrupting the operations and performance of other virtual networks. These virtual networks can also be deployed on demand and dynamically allocated. To realize these isolated virtual networks, SoftAir implements three functions: network hypervisor for high-level virtualization; wireless hypervisor at SD-BSs and switch hypervisor at SD-switches for low-level virtualization.

Remark: The detailed qualitative comparison of state-of-the-art W-SDN solutions and SoftAir is summarized in Figure 12.

WSDN	SDN Architecture	Scalability	Network virtualization	SD Traffic engineering	Research community
OpenRoads	SD-WiFi network	High	FlowVisor & SNMPVisor	No specific solution	Academia/Industry
OpenRadio	Programmable data plane	Low	No specific solution	No specific solution	Academia
CloudMAC	SD-MAC in WANs	Low	- Dynamic spectrum use - On-demand AP - Downlink scheduling	Seamless AP switch-off system	Academia
Odin	SD-MAC in WANs	Low	Hidden terminal mitigation	- Seamless mobility - Load balancing	Academia
CellSDN	SD-CN	Low	Basic concept	No specific solution	Academia/Industry
ADRENALINE	SD-CN	Low	OpenStack (customer SDN controllers)	SD Optical OFDM system	Industry
SoftRAN	SD-RAN	Low	No specific solution	Big-base station	Academia/Industry
Cloud-RAN	SD-RAN	Low	Limited exploration	Collaborative PHY operations	Academia/Industry
SK Telecom	SD-CN & SD-RAN	Low	Not supported	Collaborative PHY operations	Industry
DOCOMO	SD-RAN	Low	Limited scheme, e.g., NOMA	Collaborative PHY operations	Industry
CONTENT	SD-CN & SD-RAN	Low	Infrastructure manag. layer	Service orchestration layer	EU
SoftAir	SD-CN & SD-RAN	High	- Network hypervisor - Wireless hypervisor - Switch hypervisor	- Collaborative & coordinate processing (PHY) - Collaborative scheduling (MAC) - SD mobility management (NW)	Academia/Industry

Figure 12. The comparison of existing W-SDN solutions [3].

CHAPTER 4

SOFTAIR MANAGEMENT TOOLS

Cloud orchestration aims to automate the configuration, coordination and management of software and software interactions in the cloud environment. To support cloud orchestration in SoftAir, to enable the promising features and to maximize the overall performance of SoftAir, four essential and general management tools need to be developed: (1) in-band control traffic balancing, (2) traffic-driven optimal network planning, (3) resource-efficient network virtualization, (4) distributed and collaborative traffic classifier.

4.1 Control Traffic Balancing

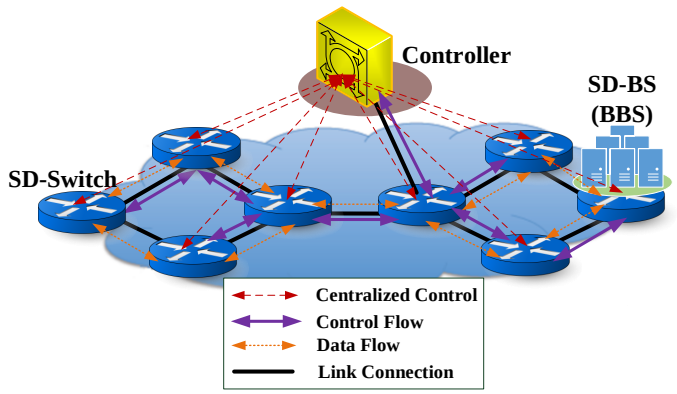
We develop an in-band control traffic balancing for a centralized controller with an objective to find the optimal control traffic forwarding paths for each switch/BS in such a way the average control traffic delay in the whole network is minimized [35]. An efficient algorithm, called polynomial-time approximation algorithm (PTAA), is proposed to yield the fast convergence to a near optimal solution of in-band signaling paths.

4.1.1 Motivation and Related Work

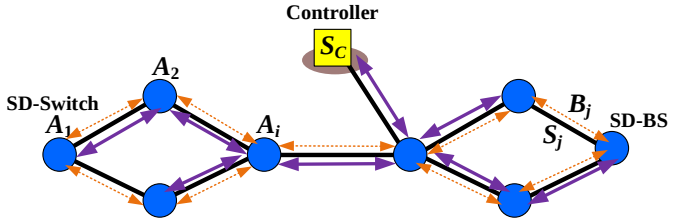
With decoupled networking architecture, the timely delivery of control messages for each SD-switch or SD-BS largely impacts the efficiency and effectiveness of SDNs, especially when in-band mode [36] is used for control traffic that could significantly affects the system performance with the combined traffic of the original data and the control messages. Thus, it becomes a great challenge to support the load-balancing of in-band control traffic for the minimum network delay via a centralized controller in SDNs. We foresee that SDN technologies would be gradually adopted in enterprises in in-band mode, as we start to address and resolve remaining technical issues. However, existing work all focuses on balancing data traffic in data plane, such as prioritizing the interactive, elastic, and background traffic in [37]. Different from data traffic balancing which aim to evenly distribute

data traffic flows among network links, control traffic balancing is much more challenging particular for in-band control [2]. It aims to find the control message forwarding paths of each switch in such a way that the control message delay can be minimized subject to acceptable performance for the original data traffic. This control traffic forwarding problem is extremely critical in SDNs because the timely delivery of control traffic initiated by Openflow switches, e.g, the first packet of every new flow and the traffic/congestion status, directly impacts the effectiveness of the routing strategies determined by the controller.

In this section, by using queueing network theory [38], we address the control traffic forwarding issue by formulating it as *a nonlinear optimization problem*. However, the complexity of such a formulation is extremely high due to (i) its nonlinearity and (ii) massive variables of link traffic assignments for large-size networks. As a result, the conventional methods for nonlinear optimization problems, such as interior point methods [39], become impractical both in terms of computation time and storage. Therefore, the principle solving method for these large-scale nonlinear problems is to find an approximate and near optimal solution in the solution space [40]. Towards this, we design a fast convergent algorithm for the control traffic balancing problem, based on the alternating direction method of multipliers (ADMM) [41], which is an emerging parallel and fast first-order method for solving large-scale convex optimization problems. In particular, we propose a polynomial-time approximation algorithm (PTAA) that applies the primal-dual update rules of ADMM approach to solve the formulated large-scale convex optimization problem. In particular, PTAA is an iterative algorithm that accurately approximates the optimal solution with fast convergence. We prove that PTAA follows the rapid convergence rate $O(1/c^m)$ with a constant $c > 1$ and iteration number m. Such fast convergence property is extremely important for SDNs because the time-varying traffic pattern in both data plane and control plane may require the fast re-planning of forwarding paths between switches and controller. Performance evaluation confirms that the proposed PTAA provides network delay for control traffic similar to the benchmarks from brute force algorithms, and outperforms



(a) Entire network architecture.



(b) Graph model with overlaid traffic flows.

Figure 13. In-band control traffic over SoftAir.

the conventional single- and multi-path solutions with at least 80% delay reduction that is time-efficient and could be executed in parallel. To the best of our knowledge, this work is the first to address control traffic balancing problem in SDNs along with the provably fast-convergent algorithm to yield the near optimal solution.

4.1.2 System Model and In-Band Control Traffic

Given SoftAir architecture, one key SDN feature is to separate the data plane from the control plane. With such decoupled networking architecture, the timely delivery of control messages for each SD-switch or SD-BS (i.e., BBS) largely impacts the efficiency and effectiveness of SoftAir. Moreover, it is cost-prohibitive to deploy large-scale SDN with out-band signaling where each hardware device is directly connected to the controller with a separate control channel. We foresee that SDN technologies would be gradually adopted in enterprises in in-band mode [36] with the combined traffic of the original data and the control messages. Hence, we aim to support the load-balancing of in-band control traffic for the minimum network delay via a centralized controller in SoftAir. A typical network

topology of SoftAir as shown in Figure 43 generally consists of multiple Openflow enabled switches or BSs, which constitutes data plane, and a centralized network controller. It is modeled by a network graph $G = (V, J)$ in Figure 13(b), where V is a set of SD-switches/SD-BSs with total n devices (i.e., $|V| = n$) and J is a set of links with total $|J|$ links. Moreover, the traffic model for the in-band signaling is considered. Without loss of generality, both control and data flows are modeled by regenerative traffic with the regenerative service processes for both link transmission and the controller's serving capability. In particular, the control traffic of each switch i is modeled by a regenerative arrival process A_i with the mean value σ_i. For the jth link and $j \in J$, the existing data flow follows a regenerative arrival process B_j with the mean value λ_j and the link serving time S_j follows another regenerative process with the mean time $1/\mu_j$. Moreover, the optimal centralized controller i^*, whose location can be determined as the one that minimizes the control message delay, has the serving capability with the mean time $1/\mu_C$.

4.1.3 Control Traffic Balancing Problem

With traffic models of data and control flows, we provide a load-balancing scheme that balances link traffic loads of the additional in-band control flows to minimize the link transmission delay. Specifically, the traffic assignment matrix $\mathbf{x} = [x_{ij}]_{i \in V, j \in J}$, where x_{ij} denotes the amount of control traffic that the ith device (i.e., switch or BS) contributes to the jth link, is obtained with respect to minimizing the average network delay over the network. To achieve load balancing, multi-path routing is adopted, where given P_i as a set of available paths for the ith device and $i \in V$, this device can forward the control messages to the controller via $|P_i|$ available paths. To characterize possible multi-path routings of control flows, for the flow from the ith device, we define a topology matrix $\mathbf{T}_i$ of size $|J| \times |P_i|$ as follows:

$$\mathbf{T}_i[j, p] = \begin{cases} 1, & \text{if the } j\text{th link lies on the } p\text{th path;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The matrix $\mathbf{T}_i$ maps the traffic from paths to links and should always be full column-rank to avoid redundant paths. Its left-inverse matrix $\mathbf{T}_i^{-1} = [t_{i1}, t_{i2}, ..., t_{i|J|}]$ exists and has the size $|P_i| \times |J|$, where t_{ij} is the column vector that maps the jth link to all possible paths of the ith device's flow. t_{ij} is obtained by multiplying $\mathbf{T}_i^{-1}$ with the jth standard basis e_j, i.e., $t_{ij} = \mathbf{T}_i^{-1} e_j$. While each device i brings a control flow with the mean value σ_i, the device i^*, where the controller is directly attached, can send its flow to controller without going through the network (i.e., $x_{i^*j} = 0, \forall j \in J$). We set up the equalities for the control flow conservation of devices as $\|\mathbf{T}_i^{-1}[x_{i1}, ..., x_{i|J|}]^T\|_1 = \sigma_i, \forall i \in \widetilde{V} := V \setminus \{i^*\}$, where $\|\cdot\|^T$ and $\|\cdot\|_1$ denote the transpose and 1-norm of vector, respectively. Let $d_{ij} = \|\mathbf{T}_i^{-1} e_j\|_1$, such equalities can be further simplified as

$$\sum_{j \in J} d_{ij} x_{ij} = \sigma_i \quad \forall\, i \in \widetilde{V}, \tag{2}$$

which is the flow conservation constraint, implying that the control flow initiated by each device is split into multiple outgoing flows on the selected transmission links. Furthermore, with the aid of *Little's law* [38], the average network delay D over the network for the control messages is obtained as

$$D = \frac{1}{\sum\limits_{i \in \widetilde{V}} \sigma_i + \sum\limits_{j \in J} \lambda_j} \sum_{j \in J} \frac{\sum\limits_{i \in \widetilde{V}} x_{ij} + \lambda_j}{\mu_j - (\sum\limits_{i \in \widetilde{V}} x_{ij} + \lambda_j)}. \tag{3}$$

In particular, for link $j \in J$, new packets arrive with rate $(\sum_{i \in \widetilde{V}} x_{ij} + \lambda_j)$ and stay an average time of $1/[\mu_j - (\sum_{i \in \widetilde{V}} x_{ij} + \lambda_j)]$. Summing queue backlogs over all links, the average network delay is thus yielded, as the total external arrivals of control and data traffic into the network are $(\sum_{i \in \widetilde{V}} \sigma_i + \sum_{j \in J} \lambda_j)$. In addition, to balance the traffic loads among all links, every link should have finite transmission delay. From the formulation in (3), such finite link delay conditions are equivalent to

$$\sum_{i \in \widetilde{V}} x_{ij} < \mu_j - \lambda_j \quad \forall\, j \in J, \tag{4}$$

which ensure the incoming traffic rates are less than the link service rates and link delays remain nonnegative. Therefore, with the above accomplishments, we define the Control Traffic Load-Balancing Problem as follows.

Definition 1 ***[Control Traffic Load-Balancing Problem.]*** *Given a SoftAir system modeled by $G = (V, J)$ with the controller location $i^* \in V$, control traffic arrival rates σ_i, a set of topology matrices $\boldsymbol{T}_i$, $\forall i \in V$, data traffic rates λ_j, and link serving rates μ_j, $\forall j \in J$, the load-balancing optimization problem to be solved by the controller is*

$$\begin{array}{ll} \textbf{\textit{Minimize}} & D\left(x_{ij}; \forall\, i \in \widetilde{V} = V \setminus \{i^*\},\ j \in J\right) \\ \textbf{\textit{Subject to}} & (2) \textit{ and } (4) \end{array} . \tag{5}$$

4.1.4 Polynomial-Time Approximation Algorithm (PTAA)

To exploit ADMM [41] for the proposed load-balancing optimization problem, there are two steps as follows. We first formulate the dual problem from the given primal problem. We then alternatively solve both problems for the optimal solution.

Theorem 1 *The dual problem of (1) is as follows:*

$$\begin{array}{ll} \textbf{\textit{Find:}} & x_{ij} \textit{ and } \beta_{ij} \quad \forall\, i \in \widetilde{V}, j \in J \\ \textbf{\textit{Maximize}} & \frac{-1}{\sum\limits_{i\in\widetilde{V}} \sigma_i + \sum\limits_{j\in J} \lambda_j} \sum\limits_{j\in J} \frac{\sum\limits_{i\in\widetilde{V}} \beta_{ij} + \lambda_j}{\mu_j - (\sum\limits_{i\in\widetilde{V}} \beta_{ij} + \lambda_j)} \\ \textbf{\textit{Subject to}} & \begin{cases} \beta_{ij} = x_{ij} \qquad \forall\, i \in \widetilde{V}, j \in J \\ \sum\limits_{j\in J} d_{ij}\beta_{ij} = \sigma_i \qquad \forall\, i \in \widetilde{V} \\ (4) \end{cases} \end{array} . \tag{6}$$

Given (6) and the *penalty parameter* $\rho > 0$ for the *augmented Lagrangian* [41], we consider the update rules for primal variables x_{ij}, β_{ij} and dual variables γ_{ij}, $\forall i \in \widetilde{V}, j \in J$. For x-update, the following iteration is obtained: $\mathbf{x}^{(m+1)} := \underset{(4)}{\arg\min} \frac{\rho}{2} \sum\limits_{i\in\widetilde{V}} \sum\limits_{j\in J} (x_{ij} - \beta_{ij}^{(m)} + \gamma_{ij}^{(m)})^2$. To simplify the calculation, let $\hat{x}_j = \sum_{i\in\widetilde{V}} x_{ij}/(n-1)$, $\hat{\beta}_j^{(m)} = \sum_{i\in\widetilde{V}} \beta_{ij}^{(m)}/(n-1)$, and $\hat{\gamma}_j^{(m)} = \sum_{i\in\widetilde{V}} \gamma_{ij}^{(m)}/(n-1)$. Then, $x_{ij} = \beta_{ij}^{(m)} - \gamma_{ij}^{(m)} + \hat{x}_j - \hat{\beta}_j^{(m)} + \hat{\gamma}_j^{(m)}$ and the x-**update** of ADMM for

the primal problem (1) and the corresponding dual problem (6) is

$$\begin{aligned} \textbf{Find:} \quad & \hat{x}_j \quad \forall\ j \in J \\ \textbf{Minimize} \quad & \tfrac{(n-1)\rho}{2} \sum_{j\in J} (\hat{x}_j - \hat{\beta}_j^{(m)} + \hat{\gamma}_j^{(m)})^2 \quad . \\ \textbf{Subject to} \quad & (n-1)\hat{x}_j < \mu_j - \lambda_j \quad \forall\ j \in J \end{aligned} \tag{7}$$

For β-update, the iteration of $\boldsymbol{\beta}^{(m+1)}$ is $\underset{\sum_{j\in J} d_{ij}\beta_{ij} = \sigma_i\ \forall\ i\in\widetilde{V}}{\arg\min} \frac{1}{\sum_{i\in\widetilde{V}}\sigma_i + \sum_{j\in J}\lambda_j} \sum_{j\in J} \frac{\sum_{i\in\widetilde{V}}\beta_{ij}+\lambda_j}{\mu_j - (\sum_{i\in\widetilde{V}}\beta_{ij}+\lambda_j)} + \frac{\rho}{2}\sum_{i\in\widetilde{V}}\sum_{j\in J}(\beta_{ij} - x_{ij}^{(m+1)} - \gamma_{ij}^{(m)})^2$, and $\beta_{ij} = x_{ij}^{(m+1)} + \gamma_{ij}^{(m)} + \hat{\beta}_j - \hat{x}_j^{(m+1)} - \hat{\gamma}_j^{(m)}$. To rewrite β-update in terms of $\hat{\beta}_j$, $\forall j \in J$ as usual, we deal with the constraint functions by matrix operation and parameter rearrangement: $\sum_{j\in J} d_{ij}\beta_{ij} = \sigma_i \Rightarrow (n-1)\sum_{j\in J} d_{ij}\hat{\beta}_j = \sigma_i + \sum_{j\in J}(d_{ij}\sum_{l\in\widetilde{V}, l\neq i}\beta_{lj}) \Rightarrow (n-1)\sum_{j\in J} d_{ij}\hat{\beta}_j = \sigma_i + \sum_{l\in\widetilde{V}, l\neq i} f_i^l\sigma_l \quad \forall\ i \in \widetilde{V}$, where $f_i^l = \begin{pmatrix} d_{i1} & \cdots & d_{i|J|} \end{pmatrix}\begin{pmatrix} d_{l1} & \cdots & d_{l|J|} \end{pmatrix}^{\dagger}$ and $(\cdot)^{\dagger}$ denotes the pseudo-inverse of matrix. The **β-update** of ADMM is then obtained by

$$\begin{aligned} \textbf{Find:} \quad & \hat{\beta}_j \quad \forall\ j \in J \\ \textbf{Minimize} \quad & \tfrac{1}{\sum_{i\in\widetilde{V}}\sigma_i + \sum_{j\in J}\lambda_j} \sum_{j\in J} \tfrac{(n-1)\hat{\beta}_j + \lambda_j}{\mu_j - ((n-1)\hat{\beta}_j + \lambda_j)} + \tfrac{(n-1)\rho}{2}\sum_{j\in J}(\hat{\beta}_j - \hat{x}_j^{(m+1)} - \hat{\gamma}_j^{(m)})^2 \quad . \\ \textbf{Subject to} \quad & \sum_{j\in J} d_{ij}\hat{\beta}_j = \tfrac{\sigma_i + \sum_{l\in\widetilde{V}, l\neq i} f_i^l\sigma_l}{n-1} \quad \forall\ i \in \widetilde{V} \end{aligned} \tag{8}$$

Finally, the iteration of **dual-update** of ADMM is obtained:

$$\gamma_{ij}^{(m+1)} := \gamma_{ij}^{(m)} + x_{ij}^{(m+1)} - \beta_{ij}^{(m+1)} \Rightarrow \hat{\gamma}_j^{(m+1)} = \hat{\gamma}_j^{(m)} + \hat{x}_j^{(m+1)} - \hat{\beta}_j^{(m+1)}\ \forall\ i \in \widetilde{V}, j \in J \tag{9}$$

With the above accomplishments, we propose PTAA in Algorithm 1 through update rules of ADMM to solve the control traffic balancing problem. Given that the objective function of (1) is strictly convex, the linear convergence of proposed PTAA can also be obtained as: the proposed PTAA in Algorithm 1 converges to the optimal solutions with rate $O(1/c^m)$, where $c > 1$ is a constant and m is the number of iterations.

4.1.5 Performance Evaluation

We evaluate the proposed PTAA and compare it with single- and multi-path forwarding as well as the benchmark. The rapid convergence rate of PTAA follows our analytical

Algorithm 1: Polynomial-time Approximation Algorithm (PTAA)

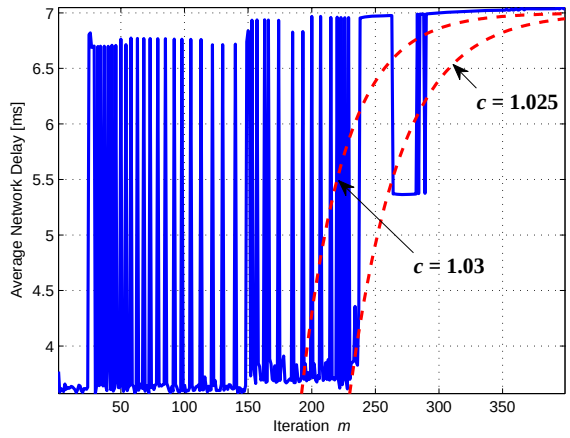
Input : Primal (1) and dual (6) problems.
Output: $x_{ij}, \forall\ i \in \widetilde{V}, j \in J$

Set $\hat{x}_j^{(0)} = 0, \hat{\beta}_j^{(0)} = 0, \hat{\gamma}_j^{(0)} = 0, \forall\ j \in J$
for $m = 0, 1, \ldots$ **do**
  Compute $\hat{x}_j^{(m+1)}, \forall\ j \in J$ according to (7)
  Compute $\hat{\beta}_j^{(m+1)}, \forall\ j \in J$ according to (8)
  Compute $\hat{\gamma}_j^{(m+1)}, \forall\ j \in J$ according to (9)
  Set $x_{ij}^{(m+1)}$ from $\hat{x}_j^{(m+1)}, \forall\ i \in \widetilde{V}, j \in J$
end

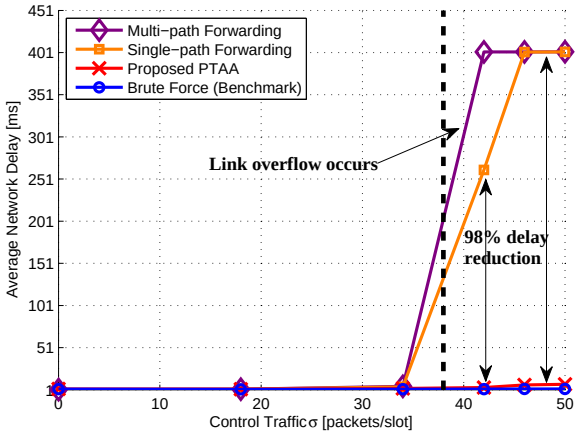
derivation as $O(1/c^m)$ shown in Figure 14(a). It provides the satisfactory values after 300 iterations which serves as a desired stopping point. Regarding traffic statistics in Internet2 OS3E [4], the data arrival rate λ_j and serving rate μ_j over links $j \in J$ are set as 200 and 1000 [packets/slot], respectively. Figs. 14(b)-14(c) shows that PTAA always has lower delay than single- and multi-path solution, and closes to the benchmark. With increasing control traffic from switches, both single- and multi-path solutions bring dramatic delay increase due to the occurrence of link overflow; however, PTAA can tolerate such higher loads through distributing extra control traffic over links with lighter data loads. Figure 14(c) further shows that PTAA can maintain network delay lower than the latency of switch processing for less control traffic, and has slightly increased delay for increasing traffic. These confirm that PTAA achieves remarkable delay reduction via a fast and parallel computation approach, thus favored by practical implementation in large-scale SoftAir.

4.1.6 Highlights

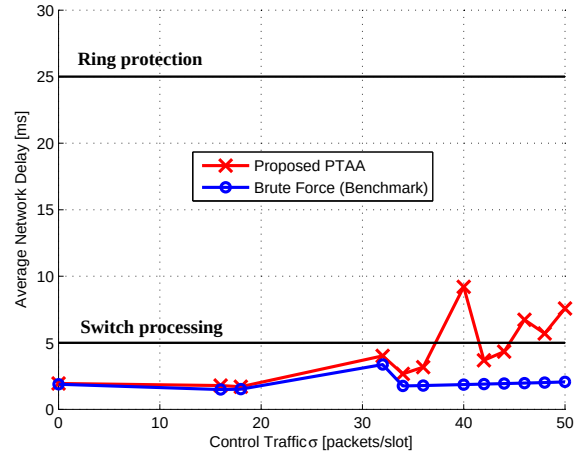
Load-balancing of in-band control traffic is addressed and the proposed PTAA solves the balancing problem in an efficient and parallel manner. Performance evaluation confirms that PTAA successfully demonstrates communication efficiency with at least 80% delay reduction via a fast and low complexity approach. We have presented a novel paradigm to facilitate on-line configurations of centralized controller in practical SDN implementations.



(a)



(b)



(c)

Figure 14. (a) Linear-fast convergence of the proposed PTAA in Algorithm 1 with rate $O(1/c^m)$. (b)-(c) Average network delay in Internet2 OS3E with 27 nodes and 36 links [4] with respect to control traffic.

4.2 Optimal Network Planning

We investigate an optimal network planning with multiple controllers that jointly optimizes controller placement and control traffic forwarding [42]. By partitioning the NP-hard planning into two sub-problems, i.e., multi-controller placement (MCP) and control traffic balancing for multiple controllers (CTB-MC), we solve them with fast-convergent algorithms.

4.2.1 Motivation and Related Work

Because of limited computational capacity of a controller, creating scalable and efficient SDN with a single controller in SoftAir is challenging for large-scale networks. The placement of multiple controllers across the entire network can address the performance limitation while retaining control centralization. In this case, several fundamental network planning problems have to be solved: the minimum number of controllers and their optimal deployment locations, control domain assignments between switches/BSs and controllers, and the corresponding optimal forwarding paths for control traffic. However, existing solutions all concern controller placement alone without involving control traffic forwarding. Lately, novel work in fast failover [43] considers both resilience-aware controller placement and control traffic routing to improve the resiliency of SDNs. In particular, exhaustive and greedy search algorithms are provided for controller placement; conventional routing tree in which each switch has only one path to the controller is adopted for control traffic forwarding. However, single-controller SDNs with in-band connections in [43] are limited for a large amount of control traffic. Moreover, no load balancing on control traffic exists in the literature, losing the capability of multi-path control traffic forwarding. Towards this, different from all existing solutions, we jointly investigate multi-controller placement and control traffic balancing for the design of optimal network planning and develop an efficient, adaptive control scheme that guarantees the optimum solution with fast replanning of controller placement and forwarding paths over time-varying QoS requirements, traffic statistics, and network topology in SDNs.

4.2.2 System Model

Given a network topology of SoftAir as $G = (V, J)$, we denote the set of SDN controllers $K \subseteq V$ with total C controllers, and the serving time capability of the kth controller is modeled as an exponential distribution with mean time value $1/\mu_C^k$. The control traffic of each device i is modeled by a regenerative arrival process A_i with mean σ_i. For the jth link, the existing data flow follows a regenerative arrival process B_j with mean λ_j, and link serving time S_j follows another regenerative process with mean time $1/\mu_j$. We address the traffic-driven design of multi-controller planing with two objectives: (i) minimize the required controllers to reduce infrastructure cost via an efficient placement and (ii) minimize the link transmission delay via an optimized traffic scheduling. $\{y_k, \forall k \in V\}$ denotes the controller locations as y_k equals one if a controller picks device k's location, and the number of total controllers C becomes $\sum_{k \in V} y_k$. $\{z_{ik}, \forall i \in V, k \in V\}$ denotes the control domain assignments between devices and controllers as z_{ik} equals one if device i is assigned to controller k. To avoid the long-distance assignments for inefficient control message transmissions, indicator variables $\{I_{ik}, \forall i \in V, k \in V\}$ are introduced to enable the localized domain assignments. I_{ik} equals one if $Distance(i, k) \leq dist$, where $Distance(i, k)$ denotes the distance between device i and controller k, and $dist$ is the predetermined value. From these variable definitions, $y_k = 1 - \prod_{i \in V}(1 - z_{ik} I_{ik})$ and

$$y_k \geq z_{ik} I_{ik} \quad \forall\ i \in V, k \in V. \tag{10}$$

To let each device be assigned to a dedicated controller, the following constraint is given as

$$\sum_{k \in V} z_{ik} I_{ik} = 1 \quad \forall\ i \in V. \tag{11}$$

Moreover, to avoid the infinite delay over controllers' incoming queues, serving capability of each controller should be enough for the arrival control messages of the assigned devices:

$$\sum_{i \in V} \sigma_i z_{ik} I_{ik} < \mu_C^k \quad \forall\ k \in K \subseteq V. \tag{12}$$

Regarding multi-controller scenarios, the traffic assignment matrix $\mathbf{x} = [x_{ij}^k]_{i\in V, j\in J, k\in V}$, where x_{ij}^k denotes the amount of control traffic on link j that originates from device i to controller k, is obtained with respect to minimizing the network delay. Similarly, flow conservation and finite link delay conditions can be respectively obtained for multiple controllers as

$$\sum_{j\in J} d_{ij}^k x_{ij}^k = \sigma_i z_{ik} I_{ik} \quad \forall\, i \in \widetilde{V}, k \in K, \tag{13}$$

$$\sum_{i\in\widetilde{V}} \sum_{k\in K} x_{ij}^k < \mu_j - \lambda_j \quad \forall\, j \in J, \tag{14}$$

where $d_{ij}^k = \|\mathbf{T}_{ik}^{-1} e_j\|_1$ from the corresponding topology matrix $\mathbf{T}_{ik}$. We have the following.

Definition 2 ***[Optimal Network Planning Problem.]*** *Given SoftAir modeled by $G = (V, J)$ with multi-controllers $k \in K \subseteq V$, control traffic rates σ_i, topology matrices $\boldsymbol{T}_{ik}$, $\forall i \in V, k \in K$, data traffic rates λ_j, and link serving rates μ_j, $\forall j \in J$, the planning optimization is*

$$\begin{array}{ll} \textbf{\textit{Find:}} & x_{ij}^k, y_k, z_{ik} \quad \forall\, i \in V, j \in J, k \in V \\ \textbf{\textit{Minimize}} & C = \sum_{k\in V} y_k \\ \textbf{\textit{Minimize}} & D_{ave}(x_{ij}^k) \\ \textbf{\textit{Subject to}} & (10), (11), (12), (13), (14) \end{array} . \tag{15}$$

where D_{ave} denotes the average network delay. This planning problem belongs to a mixed integer and continuous two-objective optimization, and the optimal values is very complicated to solve in a time-efficient manner. To provide fast solving strategy for this complex framework, we divide the original problem into two successive sub-problems as follows:

MCP Problem		***CTB-MC Problem***	
Find:	$y_k, z_{ik} \quad \forall\, i \in V, k \in V$	**Find:**	$x_{ij}^k \quad \forall\, i \in \widetilde{V}, j \in J, k \in K$
Minimize	$C = \sum_{k\in V} y_k$	**Minimize**	$D_{ave}(x_{ij}^k)$
Subject to	$(10), (11), (12)$	**Subject to**	$(13), (14)$

. (16)

4.2.3 Optimal Multi-Controller Placement via Randomized Rounding

Considering MCP problem, this integer programming problem IP_{MCP} is also linear [44]. By relaxing variables $y_k, z_{ik} \in \{0, 1\}$ to $y_k, z_{ik} \in [0, 1]$, we get the relaxed linear programming

Algorithm 2: Randomized Rounding for MCP

Input : MCP problem in Eq. (16).
Output: $(\bar{C}; \bar{y}_k, \bar{z}_{ik})$ % Optimal controller placement

Solve $\mathrm{LP_{MCP}}$. Let (y'_k, z'_{ik}) be the optimum solution.
$\bar{z}_{ik} \leftarrow 0, \forall i \in V, k \in V$
while $t \leq \log(n) + 2$ **do**
    $\bar{z}_{ik} \leftarrow 1$ with probability $\bar{p}_{ik} = z'_{ik}$
    $t \leftarrow t + 1$
end
repeat
    line 3-6
until $\sum_{i \in V} \sigma_i z_{ik} I_{ik} < \mu_C^k, \forall\, k \in V$ and $\bar{C} \leq \alpha C'$, where $y'_{max} := \max_{k \in V} y'_k$ and $\alpha = \log_{1/y'_{max}} 4C'$;

problem $\mathrm{LP_{MCP}}$. That is, it follows $\mathrm{IP_{MCP}}$ along with $y_k, z_{ik} \in [0, 1]$, $\forall i \in V, k \in V$. The solution of $\mathrm{LP_{MCP}}$ provides the optimal solution of $\mathrm{IP_{MCP}}$. Given an MCP instance modeled by $\mathrm{IP_{MCP}}$, Algorithm 9 is proposed to solve such an integer programming problem.First, the relaxed linear programming problem $\mathrm{LP_{MCP}}$ is solved to get an optimal fractional solution (OPT), denoted as y'_k, z'_{ik}, $\forall i \in V, k \in V$. Next, these fractional solutions are rounded to integer values, denoted as $\bar{y}_k, \bar{z}_{ik}$, $\forall i \in V, k \in V$, via a randomized rounding procedure [45]. The result of proposed algorithm for $\mathrm{IP_{MCP}}$ is given as follows: Algorithm 9 yields a solution of $O(\log n)$OPT with high probability, given OPT as the optimal solution of MCP problem.

4.2.4 Control Traffic Balancing for Multiple Controllers

Similarly, we adopt ADMM [41] to exploit a fast solving approach for CTB-MC problem.

Theorem 2 *Base on the results of MCP (i.e., $(\bar{C}; \bar{y}_k, \bar{z}_{ik})$), the dual problem of CTB-MC is*

$$
\begin{aligned}
&\textbf{\textit{Find:}} && x_{ij}^k \text{ and } \beta_{ij}^k \quad \forall\, i \in \widetilde{V}, j \in J, k \in K \\
&\textbf{\textit{Maximize}} && \frac{-1}{\sum_{i \in \widetilde{V}} \sigma_i + \sum_{j \in J} \lambda_j} \sum_{j \in J} \frac{\sum_{i \in \widetilde{V}} \sum_{k \in K} \beta_{ij}^k + \lambda_j}{\mu_j - (\sum_{i \in \widetilde{V}} \sum_{k \in K} \beta_{ij}^k + \lambda_j)} \\
&\textbf{\textit{Subject to}} && \begin{cases} \beta_{ij}^k = x_{ij}^k & \forall\, i \in \widetilde{V}, j \in J, k \in K \\ \sum_{j \in J} d_{ij}^k \beta_{ij}^k = \sigma_i \bar{z}_{ik} I_{ik} & \forall\, i \in \widetilde{V}, k \in K \end{cases} \\
& && (14)
\end{aligned}
\tag{17}
$$

Given Eq. (17) and the penalty parameter $\rho > 0$, we consider the update rules for primal variables x_{ij}^k, β_{ij}^k and dual variables γ_{ij}^k, $\forall i \in \widetilde{V}, j \in J, k \in K$. For x-update, the following iteration is obtained: $\mathbf{x}^{(m+1)} := \underset{(14)}{\arg\min} \frac{\rho}{2} \sum_{i\in\widetilde{V}} \sum_{j\in J} \sum_{k\in K} (x_{ij}^k - \beta_{ij}^{k(m)} + \gamma_{ij}^{k(m)})^2$. To simplify the calculation, let $\widetilde{n} = n - |\{z_{kk} = 1, \forall k \in K\}|$, $\check{x}_j = \sum_{i\in\widetilde{V}} \sum_{k\in K} x_{ij}^k / \widetilde{n}\bar{C}$, $\check{\beta}_j^{(m)} = \sum_{i\in\widetilde{V}} \sum_{k\in K} \beta_{ij}^{k(m)} / \widetilde{n}\bar{C}$, $\check{\gamma}_j^{(m)} = \sum_{i\in\widetilde{V}} \sum_{k\in K} \gamma_{ij}^{k(m)} / \widetilde{n}\bar{C}$. Then, $x_{ij}^k = \beta_{ij}^{k(m)} - \gamma_{ij}^{k(m)} + \check{x}_j - \check{\beta}_j^{(m)} + \check{\gamma}_j^{(m)}$ and the x-**update** of ADMM for the primal CTB-MC problem and the corresponding dual problem (17) is

$$\begin{aligned} \textbf{Find:} \quad & \check{x}_j \quad \forall\ j \in J \\ \textbf{Minimize} \quad & \tfrac{\widetilde{n}\bar{C}\rho}{2} \sum_{j\in J} (\check{x}_j - \check{\beta}_j^{(m)} + \check{\gamma}_j^{(m)})^2 \quad . \\ \textbf{Subject to} \quad & \widetilde{n}\bar{C}\check{x}_j < \mu_j - \lambda_j \quad \forall\ j \in J \end{aligned} \tag{18}$$

For β-update, the iteration of $\boldsymbol{\beta}^{(m+1)}$ is $\underset{\sum_{j\in J} d_{ij}^k \beta_{ij}^k = \sigma_i \bar{z}_{ik} I_{ik}\ \forall\ i\in\widetilde{V}, k\in K}{\arg\min} \frac{\sum_{j\in J} \frac{\sum_{i\in\widetilde{V}} \sum_{k\in K} \beta_{ij}^k + \lambda_j}{\mu_j - (\sum_{i\in\widetilde{V}} \sum_{k\in K} \beta_{ij}^k + \lambda_j)}}{\sum_{i\in\widetilde{V}} \sigma_i + \sum_{j\in J} \lambda_j} + \frac{\rho}{2} \sum_{i\in\widetilde{V}} \sum_{j\in J} \sum_{k\in K} (\beta_{ij}^k - x_{ij}^{k(m+1)} - \gamma_{ij}^{k(m)})^2$, and $\beta_{ij}^k = x_{ij}^{k(m+1)} + \gamma_{ij}^{k(m)} + \check{\beta}_j - \check{x}_j^{(m+1)} - \check{\gamma}_j^{(m)}$. We deal with the constraint functions by matrix operation and parameter rearrangement as follows. Given the intermediate variable $\hat{\beta}_j = \sum_{i\in\widetilde{V}} \sum_{k\in K} d_{ij}^k \beta_{ij}^k / \widetilde{n}\bar{C}$, the following equations hold $\sum_{i\in\widetilde{V}} \sum_{k\in K} d_{ij}^k \beta_{ij}^k = \widetilde{n}\bar{C}\hat{\beta}_j$; $\sum_{i\in\widetilde{V}} \sum_{k\in K} \beta_{ij}^k = \widetilde{n}\bar{C}\check{\beta}_j$. We further have $\check{\beta}_j = g_j^1 \hat{\beta}_j + g_j^2 \check{\beta}_j$, where g_j^1 and g_j^2 is from $\begin{pmatrix} g_j^1 & g_j^2 \end{pmatrix} = \begin{pmatrix} [1\cdots1] & \cdots & [1\cdots1] \end{pmatrix} \begin{pmatrix} [d_{1j}^1 \cdots d_{1j}^{\bar{C}}] & \cdots & [d_{\widetilde{n}j}^1 \cdots d_{\widetilde{n}j}^{\bar{C}}] \\ [1\cdots1] & \cdots & [1\cdots1] \end{pmatrix}^\dagger$ and $(\cdot)^\dagger$ denotes the pseudo-inverse of matrix. The β-**update** of ADMM is then obtained by

$$\begin{aligned} \textbf{Find:} \quad & \check{\beta}_j \quad \forall\ j \in J \\ \textbf{Minimize} \quad & \tfrac{1}{\sum_{i\in\widetilde{V}} \sigma_i + \sum_{j\in J} \lambda_j} \sum_{j\in J} \tfrac{\widetilde{n}\bar{C}\check{\beta}_j + \lambda_j}{\mu_j - (\widetilde{n}\bar{C}\check{\beta}_j + \lambda_j)} + \tfrac{\widetilde{n}\bar{C}\rho}{2} \sum_{j\in J} (\check{\beta}_j - \check{x}_j^{(m+1)} - \check{\gamma}_j^{(m)})^2 \quad . \\ \textbf{Subject to} \quad & \sum_{j\in J} \tfrac{1 - g_j^2}{g_j^1} \check{\beta}_j = \tfrac{\sum_{i\in\widetilde{V}} \sum_{k\in K} \sigma_i \bar{z}_{ik} I_{ik}}{\widetilde{n}\bar{C}} \end{aligned} \tag{19}$$

Finally, the iteration of **dual-update** of ADMM is obtained:

$$\gamma_{ij}^{k(m+1)} := \gamma_{ij}^{k(m)} + x_{ij}^{k(m+1)} - \beta_{ij}^{k(m+1)} \Rightarrow \check{\gamma}_j^{(m+1)} = \check{\gamma}_j^{(m)} + \check{x}_j^{(m+1)} - \check{\beta}_j^{(m+1)}\ \forall\ j \in J. \tag{20}$$

We propose PTAA for multi-controllers in Algorithm 3 to solve CTB-MC problem.

Algorithm 3: Polynomial-Time Approximation Algorithm (PTAA) for CTB-MC

Input : CTB-MC problem in Eq. (16).
Output: $x_{ij}^{k}, \forall\ i \in \widetilde{V}, j \in J, k \in K$

Set $\check{x}_j^{(0)} = 0, \check{\beta}_j^{(0)} = 0, \check{\gamma}_j^{(0)} = 0, \forall\ j \in J$
for $m = 0, 1, \ldots$ **do**
  Compute $\check{x}_j^{(m+1)}, \forall\ j \in J$ according to Eq. (18) for D_{ave}
  Compute $\check{\beta}_j^{(m+1)}, \forall\ j \in J$ according to Eq. (19) for D_{ave}
  Compute $\check{\gamma}_j^{(m+1)}, \forall\ j \in J$ according to Eq. (20)
  Set $x_{ij}^{k(m+1)}$ from $\check{x}_j^{(m+1)}, \forall\ i \in \widetilde{V}, j \in J, k \in K$
end

4.2.5 Performance Evaluation

We evaluate the proposed solving algorithms, including Algorithm 9 for the MCP sub-problem and Algorithm 3 for the CTB sub-problem for the network replanning problem in both Internet2 OS3E network [4] and Sprint GIP backbone network [5]. As our objective is to minimize the control traffic latency (e.g., either average or maximum delay), we focus on evaluating average delay D_{ave}, while maximum delay D_{max} can be examined in a similar way. Also, we average over 30 samples for each evaluation point. In the following, we first evaluate Algorithm 9 for MCP in Internet2 OS3E and Sprint GIP backbone. On top of that, we compare Algorithm 3 with several control traffic forwarding schemes, including (i) shortest-path routing [46], (ii) the multi-path forwarding, and (iii) the benchmark. Specifically, (i) shortest-path solution [46] adopts hop-counts as routing metric and employs the shortest path strategy to guide control traffic from a switch to the controller. While adopting hop-count metric as well, (ii) the multi-path solution equally splits control traffic loads among all available next-hops of a switch and applies the shortest path strategy to guide the corresponding multiple routes. (iii) The benchmark solution is implemented that solves the CTB problem through the brute-force exhaustive search.

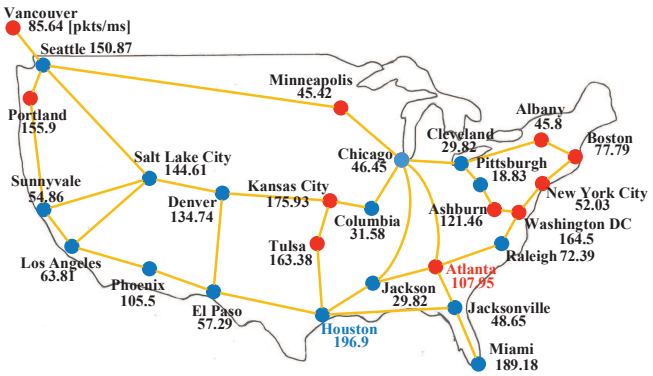


Figure 15. Optimal MCP in Houston and Atlanta; two switch groups (i.e., in red and blue) with given controller serving capability as listed.

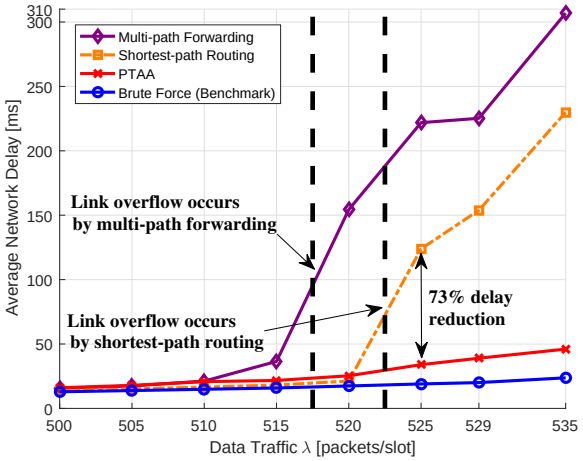


Figure 16. Average delay in Internet2 OS3E under optimal MCP in Figure 15 with respect to existing data traffic. Link serving rate $\mu_j = 600$ [pkts/ms], $\forall j \in J$ and control traffic arrival $\sigma_i = 10$ [pkts/ms], $\forall i \in V$.

Internet2 OS3E: Internet2 OS3E [4] is launched to support research and networking in the United States. Being the next-generation innovation platform, it has been selected as a representative setup for evaluating controller placement in SDNs by many studies [35, 43, 46–48]. As shown in Figure 15, this common platform has 27 nodes and 36 links and is adopted for our MCP and CTB evaluations. In particular, by first applying Algorithm 9 with given controller serving capability, Figure 15 shows that two controllers in Houston and Atlanta are selected with two switch groups. These results come from the joint consideration of traffic and topological attributes when solving MCP problem. On the other hand, in [46], hop-count is the only concerned attribute in single-controller placement, and

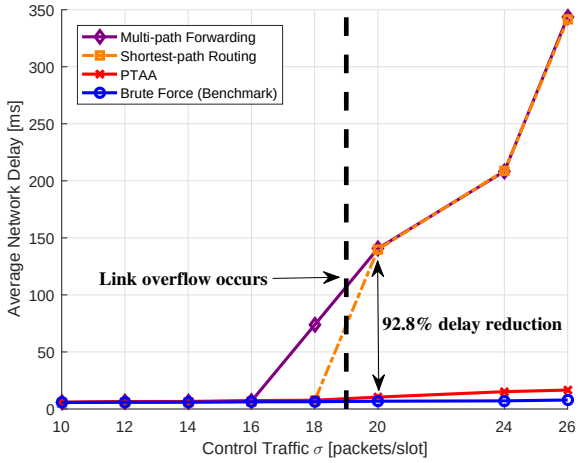


Figure 17. Average delay in Internet2 OS3E under optimal MCP in Figure 15 with respect to control traffic arrivals. Link serving rate $\mu_j = 800$ [pkts/ms] and existing data traffic rate $\lambda_j = 600$ [pkts/ms], $\forall j \in J$.

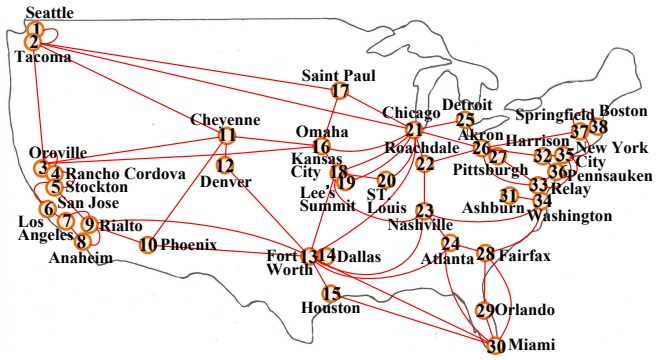
Chicago is selected as the optimal location in regard to average hop number. However, this work does not consider traffic statistics and the obtained location will no longer be optimal regarding time-varying control and data traffic.

Moreover, according to the acquired controller placement, we execute Algorithm 3 for CTB in Internet2 OS3E. Specifically, given control traffic rates 10 [pkts/ms] from switches, Figure 16 shows the average delay for CTB solutions with respect to existing data traffic. With increasing data traffic, PTAA does not occur link overflowing and outperforms shortest- and multi-path solutions with at least 73% delay reduction. Alternatively, given fixed data traffic rates 600 [pkts/ms] and increasing control traffic, Figure 17 shows that PTAA significantly reduces the delay and supports the performance close to the benchmark. The capability of balancing control traffic via the proposed PTAA is then validated.

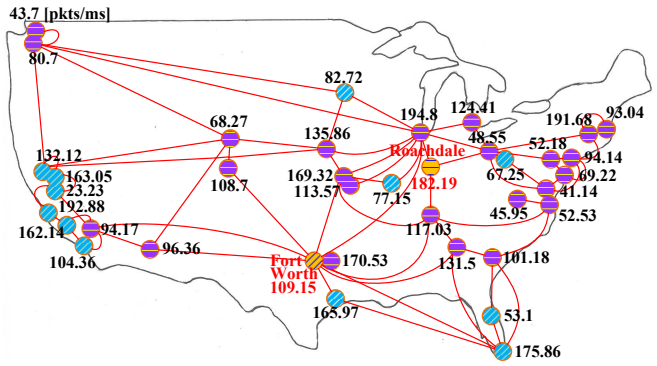
Sprint GIP Backbone: in addition to examining the proposed algorithms in a research-intensive setup, we also evaluate our algorithms in a practical network scenario: Sprint GIP backbone network [5]. Specifically, Sprint Corporation provides the real backbone network topology and the actual link delay of data traffic. Such delay information is utilized to estimate the corresponding data traffic arrival and serving rates. As shown in Figure 18(a), the GIP network topology of North America with 38 nodes and 66 links is adopted for

our evaluations. Given control traffic rate σ_i from switches, Figure 18(b) and Figure 18(c) show the optimal multi-controller placement in Sprint GIP network with respect to controller serving capabilities, in which rand denotes an uniformly distributed random variable between 0 and 1. Note that to enable localized control domain assignments, i.e., I_{ik} in Eq. (35), it is assumed that each controller can regulate its three-hop switch neighbors. With better controller-serving capability, Figure 18(b) shows that two controllers in Roachdale and Fort Worth are selected with their respective switch groups. Controllers with greater capabilities are not selected in optimal MCP, as not only controller computation capabilities but their topological attributes are concerned with the minimum controller requirement. Moreover, Figure 18(c) further shows the optimal MCP with less controller serving capability, in which three controllers in Roachdale, Lee's Summit, and Rialto are selected with their respective switch groups. Similarly, these controllers are favored to serve as traffic hubs because of their great serving capabilities and their central locations with many direct links to switches.

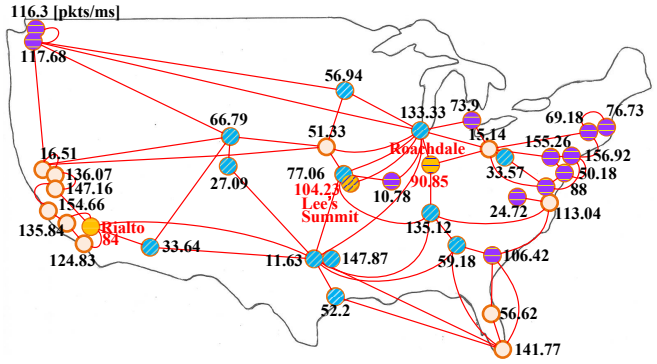
Based on the obtained controller placement via MCP, we further examine the transmission delay of control traffic under the proposed CTB solution in Algorithm 3. Apart from comparing with shortest-path scheme [46] and multi-path forwarding as before, two delay bounds relevant to today's networks [49] are considered. Specifically, (i) ring protection, 50 [ms], concerns the target restoration time of a ring topology (e.g., SONET ring). It covers the time from fault detection to when flowing traffic in the opposite direction along the ring. (ii) Shared-mesh restoration, around 200 [ms], serves as the point at which voice calls start to drop, or ATM circuit rerouting may be triggered. Given link serving rate 1000 [pkts/ms], Figure 19 shows the average delay of the proposed PTAA and of several possible solutions in the log scale. Specifically, with increasing control traffic from switches, PTAA always has lower delay, close to the benchmark delay, than other solutions. While shortest- or multi-path scheme brings a dramatic delay increase from link overflow, our solution can tolerate such high loads by distributing control traffic over links with lighter data loads.



(a)



(b)



(c)

Figure 18. (a) Sprint GIP network of North America with 38 **nodes and** 66 **links [5]. (b) Optimal MCP in 13 Fort Worth and 22 Roachdale and two respective groups with better controller-serving capability. (c) Optimal MCP in 9 Rialto, 19 Lee's Summit, and 22 Roachdale and three respective groups with less controller-serving capability.**

Moreover, with better link-serving capability (i.e., serving rate 1200 [pkts/ms]), Figure 20 shows that the proposed PTAA greatly surpasses other solutions with the delay close to the benchmark performance. In this case, our solution greatly reduces the delay, particularly

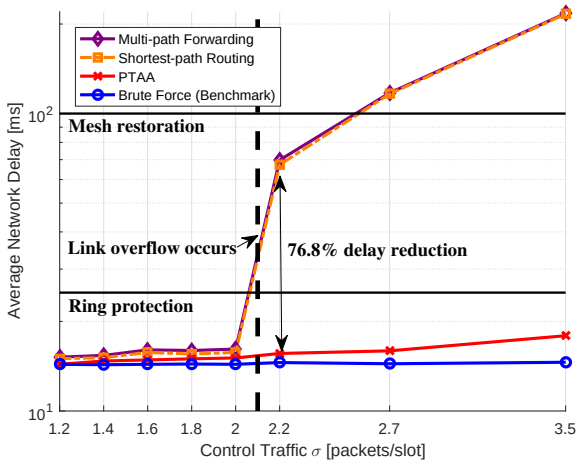


Figure 19. Average delay in Sprint GIP backbone under optimal MCP in TABLE 18(b) with respect to control traffic arrivals. Link serving rate $\mu_j = 1000$ [pkts/ms], $\forall j \in J$.

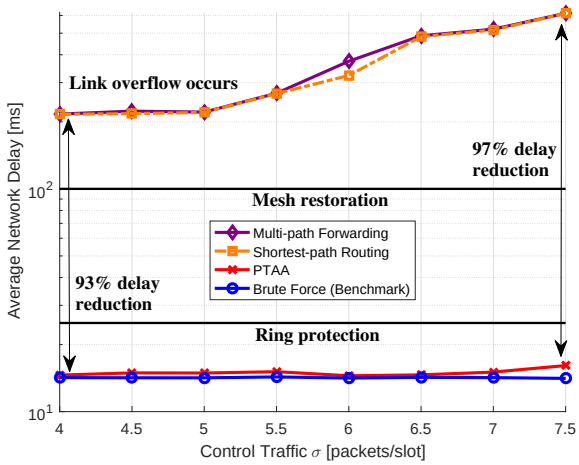


Figure 20. Average delay in Sprint GIP backbone under optimal MCP in TABLE 18(b) with respect to control traffic arrivals. Link serving rate $\mu_j = 1200$ [pkts/ms], $\forall j \in J$.

for large arrivals, while shortest- or multi-path routing induces link overflow even for limited control traffic volume. Also, similar to Figure 19, Figure 20 indicates that PTAA leads to the delay performance lower than the latency of ring protection, and retains the slightly increased delay for increasing traffic. In short, all of the above observations suggest that by employing information of traffic statistics, our solution brings better controller placement and link resource utilizations and outperforms existing schemes with at least 73% delay reduction. Yielding optimal controller placement and traffic balancing, the above results

confirm that the proposed randomized rounding provides suitable MCP in a timely manner, and the proposed PTAA significantly reduces delay via a fast and parallel computation approach, thus favored in practical implementation for multi-controller SoftAir.

4.2.6 Highlights

This study addressed traffic-driven SDN planning of in-band control traffic as a nonlinear multi-objective (mixed integer and continuous) optimization problem. This complex optimization was solved in a timely manner by partitioning the original problem into two subproblems (the placement and traffic balancing). Performance evaluations confirmed that the proposed control scheme, based on the minimum number of required controllers, demonstrated communication efficiency via a fast and low complexity approach with at least 73% delay reduction, close to the benchmark performance. Thus, a novel paradigm was successfully presented that facilitated on-line configurations of centralized multi-controllers in practical SDN implementations.

4.3 Network Virtualization

The network virtualization layer of SoftAir is designed to create a set of virtual (or logic) networks on the shared network infrastructure. The virtual networks can be dedicated (i) to different network services/applications so that each service/application can be treated with customized and independent resource provisioning algorithms, (ii) to different network operators so that multiple operators can dynamically share the same network infrastructure along with the associated spectrum and infrastructure sharing, and (iii) to facilitate the cooperation and coexistence of different technologies, e.g., multi-radio access technologies. To realize these isolated virtual networks, two functions are introduced: the network hypervisor for high-level virtualization, and the wireless hypervisor and the switch hypervisor for low-level virtualization.

4.3.1 Motivation and Related Work

During the past decade, the increment and complexity of tenants' application demands and requirements motivate the reconsideration of better traffic engineering solutions. In particular, as applications dramatically increase with various types and thus become more challenging to address, network operators in commercial clouds and data centers have been trying to improve network performance while fulfilling application requirements. However, this objective is almost impossible to accomplish with the current *closed/fixed* network architectures [50]. Exploiting the novel SDN architecture by cloud computing and data centers, a multi-tenant and resource sharing scheme is widely considered due to its infrastructure and maintenance cost-effectiveness, simplification, lower system requirements and flexibility. Specifically, in single tenant solutions, tenants' applications have their own dedicated resource and nearly 45% of these resources are idle for most of the time. On the other hand, in multi-tenant solutions, resources are shared among tenants, which implies the efficient resource utilizations. However, under such multi-tenancy scenarios, the resource assignments might be overlapped due to the sharing, and high-demanded tenants can monopolize all the shared resources, thus greatly affecting other tenants' operations [51, 52]. In that case, while SDN's global network view might allow the supervision of tenants' resource consumption to detect the high-demanded tenants, there is still a need of virtualization mechanism that isolates and prioritizes tenants' resource usages from each other, thus allowing customized performance and security level. Moreover, the existing work of traffic engineering in SDNs [50, 53] neither consider multi-tenancy scenarios with virtualization nor examine the QoS provisioning in flow allocation for various tenants' applications.

Leveraging SDN's new system architecture, it is possible to develop a routing framework in multi-tenancy environments with the provisioning of QoS-aware flow allocation and tenant isolation and prioritization, which is significant for cloud computing and enterprise data centers [54]. Therefore, in this section, a jointly optimized design of virtualization and routing is addressed and an adaptive solution is proposed to react time-varying

QoS requirements, network topologies, and traffic statistics in a real-time manner. Specifically, a fine-grained network virtualization is first proposed to slice the physical network infrastructure into several isolated subnets for multiple tenants. Ideally, the infrastructure slicing should support 100% independence among multi-tenants, i.e., no shared edges among tenants' subnets, in such a way each tenant's preference can be satisfied. Towards this, the network and switch hypervisors are introduced to efficiently give a feasible solution to the NP-complete problem of network graph partitioning [55], supporting network slicing for tenants isolation and prioritization. Furthermore, a QoS-aware dynamic flow allocation is proposed to enable fast flow allocation with regards of traffic variations and end-to-end QoS requirements. The management tool of QoS-aware Virtualization-enabled Routing (QVR) algorithm is further proposed to facilitate an adaptive feedback control of network virtualization and flow allocation and to improve service performance for achieving tenant's satisfaction, thus providing a customized solution for multi-tenancy SDNs. Performance evaluation confirms that QVR outperforms conventional approaches with less shared edges to ensure resource isolation of infrastructure usages and traffic delay for multi-tenants, successfully enabling a virtualization-enabled traffic engineering with reliable and efficient transmissions in practical implementations of multi-tenancy SDN. To the best of our knowledge, this work is the first to provide a joint consideration of network virtualization and routing decision with QoS provisioning in centralized controller SDN.

4.3.2 Joint Virtualization and Routing Decision Problem

As shown in Figure 21, our objective is to develop a routing algorithm that provides complete isolation of resource usages by multi-tenants upon the same SDN infrastructure, in such a way QoS requirements of all tenants' applications are fulfilled at the same time. Towards this, it consists of two phases designs, i.e., network virtualization and QoS-aware flow allocation, and a management tool, i.e., QVR algorithm. In particular, the first phase of network virtualization sites between data and control planes, and divides the network resources and infrastructure into isolated portions to separate application flows from different

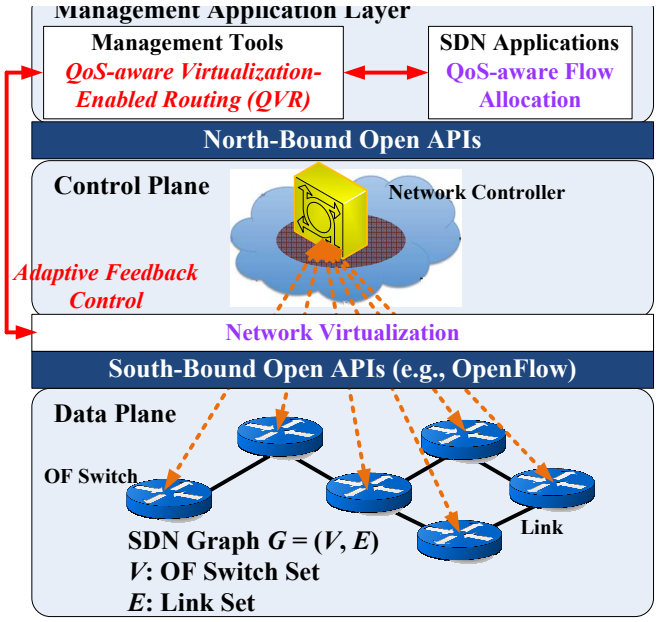


Figure 21. SDN system architecture with network virtualization.

tenants. This virtualization maintains the isolated resource portions unchanged regardless of the new flow arrivals. Furthermore, for the second phase of flow allocation, it sites at application layer, allocates flows according to their respective QoS demands, and calculates and adjusts flow paths for every new flow. Moreover, the QVR management tool combines these two phases to determine the suitable pathes for new flows as well as to decide the optimal resource allocation according to current network status. In addition, if the flow allocation phase fails to provide the required QoS performance, QVR will further feedback the operations to virtualization phase in order to enable a better network slicing. Therefore, this joint design indeed facilitates the jointly optimized virtualization and routing in SDNs.

Network Model: the SDN is modeled by a undirected network graph $G = (V, E)$ as shown in Figure 21, where V is the set of OF switches with total $|V|$ switches (more complicated models can be considered through the assumption that each node in V can be an aggregated node grouping multiple physical switches) and E is the set of links with total $|E|$ links. Let N denote the set of tenants that shares the same SDN with total $|N|$ tenants. While our objective aims to virtualize the physical network into multiple separated subnets

for isolated resource usages among tenants, we introduce some useful notations and definitions as follows. A graph $G' = (V', E')$ is a subgraph(subnet) of graph(network) G, i.e., $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. A connected graph G is a nontrivial graph that any two vertices in graph can be connected by at least one path in G. Moreover, the complement of a graph G, denoted as $\bar{G} = (V, \bar{E})$, has the same node set as G and the edges that are not included in G. A minimum spanning tree of a connected undirected graph G is a tree that includes every node using the minimal set of edges in terms of the given weight function. Note that as it might always exists a direct physical link between each source-destination pair, it is assumed that there are more than one feasible routes for each pair.

Network Resource Slicing: given multiple tenants sharing a SDN infrastructure, the goal of network resource slicing is to divide the topological resources among multi-tenants wisely, in such a way the resource usages can be isolated and each tenant's preference can be satisfied. Specifically, for $|N|$ tenants upon a SDN G, the subnets for each tenant $G_n = (V_n, E_n), \forall n \in N$, where $V_n \subseteq V$ and $E_n \subseteq E$, should be decided to minimize the overlapped edges (i.e., $|E_n \cap E_m| \rightarrow 0, \forall 1 \leq n < m \leq |N|$) and tenants' satisfactions in terms of weights $w_n^{max}, \forall n \in N$ are achieved. These weight parameters can be considered either from graph-centric perspective (e.g., the subnet diameter or size) or QoS-centric perspective (e.g., the largest allowable path delay or packet loss). In particular, we consider

$$W(G_n) < w_n^{max}, \quad \forall n \in N \tag{21}$$

where w_n^{max} denotes the largest allowable path weight of the subnet n and $W(G_n)$ denote the weight supported by the subnet with respect to all existing paths. Eq. (46) indicates the constraints of weight satisfactions for all tenants. Note that all subnet must be connected graphs, as every pair of nodes in a subnet should be able to communicate with each other.

To provide complete resource isolation among tenants, each subnet should better be the subset of the complement of the other $|N| - 1$ union graph, i.e., $G_n \subseteq \cap_{k=1,k\neq n}^{N} \bar{G}_k$. In general, this edge-disjoint requirement is hard to achieve, especially when there are many tenants sharing few edges of a SDN. Hence, a practical approach is to allow a certain degree of

dependent resource utilization by limiting the number of *shared edges* among tenants. In particular, an appropriate network slicing should support the following:

$$\min_{\{G_n;\forall n\in N\}} \sum_{1\le n<m\le|N|} |E_n \cap E_m|. \tag{22}$$

Regarding these shared edges, a flow allocation scheme, e.g., the one explained later, should be proposed to ensure each dependent tenant's application requirements are fulfilled. Specifically, the centralized controller of SDN has the capability to monitor and regulate tenants' traffic by implementing routing and resource allocation decision made by the proposed management tools; thus it can adaptively rearrange the shared edges and link capacities accordingly among dependent tenants. Moreover, to further provide the serving differentiation among multiple tenants, it is assumed that tenants' demands are less imperative as the order of tenant $n \in [1, N]$ grows. It implies tenant 1 has the highest serving priority while tenant N has the lowest.

Flow Allocation: to characterize the packet forwarding of application flows upon the given network slicing of multi-tenants, $X^n_{f,i,j}$ denotes the achievable packet data rate of link $(i, j) \in E^f_n$ for the flow f in subnet n, and λ^n_f denotes the packet arrival rate to the source node s^n_f, where $f \in F^n$ and $n \in N$. Therefore, the routing constraint becomes

$$X^n_{f,i,j} = 0, \forall (i, j) \notin E^f_n, f \in F^n, n \in N. \tag{23}$$

It implies that if a link is not sliced to a flow in subnet, the link packet rate should be set to zero for that specific flow. Moreover, given the link capacity c_{ij} for link $(i, j) \in E$, the corresponding capacity constraint is provided as

$$\sum_{n\in N} \sum_{f\in F^n} X^n_{f,i,j} \le c_{ij}. \tag{24}$$

Also, the flow conservation can be considered as the following:

$$\sum_{j;(i,j)\in E^f_n} X^n_{f,i,j} - \sum_{j;(j,i)\in E^f_n} X^n_{f,j,i} = \lambda^n_f \mathbb{I}_{\{i=s^n_f\}}, \quad \forall i \ne d^n_f, f \in F^n, n \in N, \tag{25}$$

where d_f^n denotes the destination of flow f in subnet n and $\mathbb{I}_{\{\cdot\}}$ is an indicator function that gives 1 when the event occurs and 0 otherwise. This implies that the outgoing flows of node should be equal to the incoming flows from neighbors plus flow arrival rate, if the source node is concerned. In addition to the above constraints, the objective of flow allocation is to decide an effective resource utilization for multi-tenants' applications with respect to the given link capacities. Specifically inspired by the work in [53, 56, 57], for each tenant, the minimum arrival rate among flows should be maximized from the flow allocation; thus,

$$\sum_{n \in N} \max_{\{X^n_{f,i,j}; \forall (i,j) \in E^f_n, f \in F^n\}} \min_{f \in F^n} \lambda^n_f \tag{26}$$

provides the total achievable data rate accordingly.

End-to-End QoS Provisioning: aiming at supporting a great variety of QoS requirements for tenants' applications, the four major end-to-end QoS are considered as follows. First, given the maximum tolerable delay, jitter, and packet loss for a specific flow (i.e., $D^{max}_{n,f}, J_{n,f}, p^{max}_{n,f} \forall f \in F^n, n \in N$), the QoS constraints are provided respectively as $\forall f \in F^n, n \in N$,

$$D^n_f = \sum_{(i,j) \in E^f_n} (\tfrac{1}{X^n_{f,i,j}} + D^{q,n}_{f,i}) < D^{max}_{n,f}; \tag{27}$$

$$var(D^n_f) < J_{n,f}; \tag{28}$$

$$\prod_{(i,j) \in E^f_n} p^n_{f,i,j} < p^{max}_{n,f}, \tag{29}$$

where $1/X^n_{f,i,j}$ and $D^{q,n}_{f,i}$ denote the transmission and queueing delay, respectively, $var(\cdot)$ gives the variance of the delay, and $p^n_{f,i,j}$ denotes the link packet loss. Moreover, to ensure sources' arrival rates are supportable by the links of forwarding paths, it implies that $\forall (i,j) \in E^f_n, f \in F^n, n \in N$,

$$X^n_{f,i,j} > \lambda^n_f. \tag{30}$$

Note that considering various tenants' applications, Eq. (44(e))-(30) may give different degrees of QoS requirements according to the applications. For example, in interactive

Table 1. Formulation for Jointly Virtualization and Routing.

Objective:

$$\min_{\{G_n;\forall n\in N\}} \sum_{1\le n<m\le|N|} |E_n \cap E_m| \tag{22}$$

$$\sum_{n\in N} \max_{\{X^n_{f,i,j};\forall(i,j)\in E^f_n, f\in F^n\}} \min_{f\in F^n} \lambda^n_f \tag{44(d)}$$

Subject to:

Subnet weight constraint:

$$W(G_n) < w^{max}_n, \forall n \in N \tag{46}$$

Routing constraint:

$$X^n_{f,i,j} = 0, \forall (i,j) \notin E^f_n, f \in F^n, n \in N \tag{44(a)}$$

Link capacity constraint:

$$\sum_{n\in N} \sum_{f\in F^n} X^n_{f,i,j} \le c_{ij}, \forall (i,j) \in L \tag{44(b)}$$

Flow conservation constraint:

$$\sum_{j;(i,j)\in E^f_n} X^n_{f,i,j} - \sum_{j;(j,i)\in E^f_n} X^n_{f,j,i} = \lambda^n_f \mathbb{I}_{\{i=s^n_f\}}, \quad \forall i \ne d^n_f, f \in F^n, n \in N \tag{44(c)}$$

End-to-end delay constraint:

$$D^n_f = \sum_{(i,j)\in E^f_n} \left(\frac{l^n_f}{X^n_{f,i,j}} + D^{q,n}_{f,i}\right) < D^{max}_{n,f}, \forall f \in F^n, n \in N \tag{44(e)}$$

Jitter constraint:

$$var(D^n_f) < J_{n,f}, \forall f \in F^n, n \in N \tag{28}$$

Packet loss constraint:

$$\prod_{(i,j)\in E^f_n} p^n_{f,i,j} < p^{max}_{n,f}, \forall f \in F^n, n \in N \tag{29}$$

Data rate constraint:

$$X^n_{f,i,j} > \lambda^n_f, \forall (i,j) \in E^f_n, f \in F^n, n \in N \tag{30}$$

applications, the latency is more effective to the need of real-time responses, and thus the delay and jitter constraints are more stringent than the loss constraint. On the other hand, for web browsing or emails, the jitter constraint is not applicable due to its little performance impact, whereas the throughput and loss constraints are of considerable significance. Therefore, the entire formulation for jointly virtualization and routing is well-established and TABLE 1 summarizes all the details.

4.3.3 Fine-Grained Network Virtualization

In this section, a fine-grained network virtualization is proposed to manage topological resources among multiple tenants. In particular, the high-level resource management and low-level resource scheduling are introduced by the designated network hypervisor and

switch hypervisor, respectively.

Network Hypervisor: the network slicing should be an efficient resource management that aims to minimize the number of shared edges among tenants, i.e., $|E_n \cap E_m| \rightarrow 0$. It thus provides isolated resource usages and minimizes the interference for the multi-tenants' applications. Towards this, the proposed network hypervisor in **Algorithm 4** perfectly suits the need of network virtualization, and the details are explained as follows. Given the serving priorities among tenants, i.e., tenant 1 (N) has the highest (lowest), **Algorithm 4** first deals with tenant 1 subnet, then tenant N subnet, and finally the remaining tenant subnets. Specifically, first of all, for tenant 1, a minimum spanning tree is searched within the SDN graph G using Kruskal's algorithm [58], and the found edges are added to the highest priority subnet G_1. Then, tenant 1's satisfaction is evaluated for Eq. 46 such that all source-destination pairs of application flows should have at least one path with path weight less than w_1^{max}. This path weight can be determined through a shortest path algorithm, e.g. Dijkstra's algorithm [59]. In particular, all source-destination pairs that do not satisfy the w_1^{max} constraint are first found. For each discovered pair, Dijkstra's algorithm is run over G_1, and new edges are added to guarantee the corresponding route has a weight less than w_1^{max}. The iteration continues until w_1^{max} is fulfilled in all source-destination pairs inside subnet G_1. Next, for tenant N, similar procedures are executed, except that the maximum spanning tree is considered. In particular, Kruskal's algorithm is applied over the graph $\bar{G}$ for a maximum spanning tree and the found edges are added to the lowest priority subnet G_N. Then, tenant N's satisfaction is evaluated with respect to w_N^{max}. Finally, the remaining subnets G_n, where $2 \leq n \leq N-1$ are constructed. In particular, Kruskal's algorithm is run over the graph $G \setminus (\cup_{k=1}^{n-1} G_k \cup G_N)$, the isolated nodes are found, and the edges from these isolated nodes to subnet G_n are added. If more than one possible edge for an isolated node is found, network hypervisor will choose the edge with less weight. Then, tenant's satisfaction is evaluated with w_n^{max}. Note that there might be some edges that are already included in the higher priority subnets, and these edges become *shared edges*

Algorithm 4: Network Hypervisor

Input : Topology G; subnet weights $w_1^{max}, \ldots, w_N^{max}$
Output: Subnets $G_1, \ldots, G_N$

Tenant 1 *subnet*

Run Kruskal's alg. over G % Obtain the min. spanning tree over G
Add found edges into G_1
Find source-destination pairs (u,v) with path weight larger than w_1^{max}
for *each discovered pair (u,v)* **do**
    while *path weight between (u,v)* $> w_1^{max}$ **do**
        Run Dijkstra alg. over G_1 for pair (u,v)
        Add new found edges into G_1
    end
end

Tenant N subnet

Run Kruskal's alg. over $\bar{G}$ % Obtain the max. spanning tree over G
Add found edges into G_N
Find source-destination pairs (u,v) with path weight larger than w_N^{max}
for *each discovered pair (u,v)* **do**
    while *path weight between (u,v)* $> w_N^{max}$ **do**
        Run Dijkstra alg. over G_N for pair (u,v)
        Add new found edges into G_N
    end
end

Tenant $2, \ldots, N-1$ *subnet*

for *each tenant* $n \in \{2, \ldots, N-1\}$ **do**
    Run Kruskal's alg. over $G_n = G \setminus (\cup_{k=1}^{n-1} G_k \cup G_N)$
    Find isolated nodes
    Add edges from isolated nodes into G_n % Transform G_n into a connected graph
    Find source-destination pairs (u,v) with path weight larger than w_n^{max}
    for *each discovered pair (u,v)* **do**
        while *path weight between (u,v)* $> w_n^{max}$ **do**
            Run Dijkstra alg. over G_n for pair (u,v)
            Add new found edges into G_n
        end
    end
end

among tenants' subnets. Therefore, through **Algorithm 4**, a single SDN can be well-sliced into multiple subnets while minimizing the number of shared edges.

Switch (Wireless) Hypervisor: while network hypervisor provides the sophisticated resource management among tenants, there is a need of an executor to fulfill such a management. Towards this, switch hypervisor is proposed to perform the resource scheduling and consists of two elements, i.e., queue-length based generalized processor sharing (GPS) and Flowvisor [60]. First, while the conventional GPS can easily enable each subnet to operate independently, it does not provide throughput efficiency in reallocating unused network resources (i.e., links) when some subnets are temporally idle, i.e., having all flow queues empty. This is because the sharing time calculation of conventional GPS is insensitive to the queue-length conditions. Instead, we propose a queue-length based GPS that assigns time portions proportional to queue lengths as follows. Consider for link $(i, j) \in E$, there are sets of $\mathcal{A}^c_{i,j}, \mathcal{A}_{i,j} \subseteq N$, i.e., inactive and active subnets respectively, and there are F^n application flows with $Q^n_{f,i,j}$ as the queue length of flow f in subnet n. Then, the sharing time of the subnet $n \in \mathcal{A}_{i,j}$ over link $(i, j) \in E$ is obtained by

$$\hat{t}^n_{i,j} = t^n_{i,j} + \sum_{k \in \mathcal{A}^c_{i,j}} \frac{\sum_{f \in F^n} X^n_{f,i,j} Q^n_{f,i,j} \mathbb{I}_{\{(i,j) \in E^f_n\}}}{\sum_{l \in \mathcal{A}_{i,j}} \sum_{f \in F^l} X^l_{f,i,j} Q^l_{f,i,j} \mathbb{I}_{\{(i,j) \in E^f_l\}}} t^k_{i,j}, \tag{31}$$

where $t^k_{i,j}, \forall k \in \mathcal{A}^c_{i,j}$ indicate unused time portions of inactive subnets. Next, a well-known software program, Flowvisor [60], is exploited to virtualize the network according to our designated of network and switch hypervisors. In particular, there are three modes of Flowvisor that can be utilized such as slicing by host IP or MAC address, by port number, and by the protocol type. An example of slicing by IP address is shown in Figure 22 with the original physical infrastructure and two subnets. The key idea is to determine the source and destination IP address for each shared switch. Moreover, regarding the better design flexibility, Figure 23 shows a more complicated example for a combined multi-slicing with five respective subnets. In particular, the original network is first sliced twice according to IP address of hosts connected to the switches, and then is sliced according to protocols such as SSH, HTTP, and Telnet. Hence, we have successfully provided a fine-grained

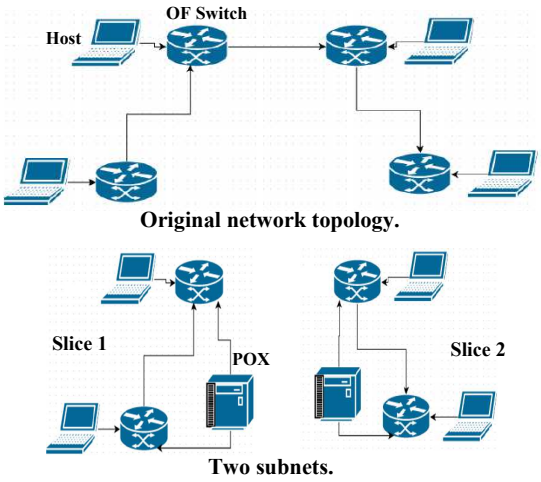


Figure 22. An example of network slicing by host IP address with two subnets.

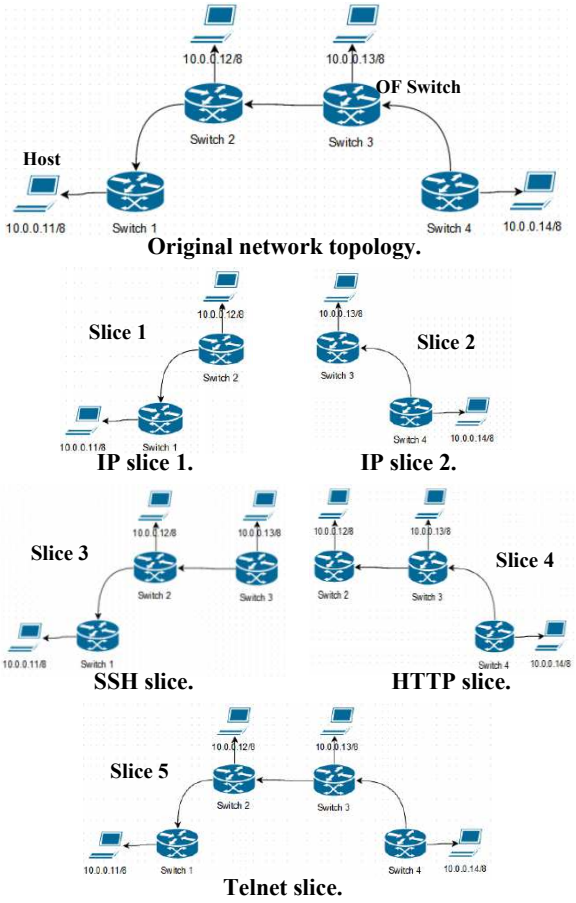


Figure 23. An example of combined multi-slicing with five subnets.

network virtualization that allows slices of multiple subnets to be defined and configured independently, facilitating the resource isolation and serving priority among tenants.

4.3.4 QoS-aware Virtualization-enabled Routing (QVR)

In this section, a management tool of QoS-aware virtualization-enabled routing (QVR) is proposed. In particular, based on the results of network virtualization, a dynamic flow allocation is first introduced to guide packet flows in such a way the flow allocation and end-to-end QoS provisioning are supported. Moreover, QVR algorithm is then proposed to combine the designated network hypervisor and QoS-aware dynamic flow allocation, thus enabling an adaptive control for a completed solution for multi-objective optimization problem in TABLE 1.

Dynamic Allocation Framework with QoS Provisioning: QoS-aware Dynamic Flow Allocation in **Algorithm 5** provides a dynamic solution to maximize the minimum flow arrival rates for multiple tenants. First of all, for each application flow in subnets, all available paths between flow source and destination are found. Next, with respect to these paths, the constraints of flow allocation and end-to-end QoS provisioning are evaluated, and the maximum achievable arrival rates are yielded accordingly. In particular, **NSQP** function in line 6 in **Algorithm 5** enables the QoS constraint evaluations that gives the true value if the QoS provisioning is not satisfied with the given arrival rate in the current round. In that case, the algorithm enables the successive round to reduce the arrival rate and finds the suitable flow allocation accordingly via line 8; otherwise, the algorithm stops and the optimal solution is reached. Finally, the maximum rate among available paths of a flow is selected as the flow arrival rate, and the minimum flow rate in a subnet and the total rates over multi-tenants can be obtained accordingly. Therefore, due to the simple operations of **Algorithm 5**, it provides a feasible solution in a timely manner, facilitating dynamic flow allocation for all tenants' applications.

QoS-aware Virtualization-Enabled Routing (QVR): QVR algorithm in **Algorithm 6** provides an adaptive feedback control between network hypervisor in **Algorithm 4** and

Algorithm 5: QoS-aware Dynamic Flow Allocation

Input : $G_1, \ldots, G_N$; link capacity $\{c_{ij}\}$; QoS indices $\{D^{max}_{n,f}, J_{n,f}, p^{max}_{n,f}\}$
Output: Total data rate λ^*; flow allocation $\{X^n_{f,i,j}\}$

for *each subnet* G_n **do**
  for *each flow* $f \in F^n$ **do**
    Find all paths between (s^n_f,d^n_f)
    for *each path* k **do**
      Initialize $\lambda^n_{f,k} = \infty$
      while ***NSQP****(Eq. (7)-(10),*$\lambda^n_{f,k}$*)* **do**
        $\lambda^n_{f,k} \leftarrow \lambda^n_{f,k} - 1$
        Find $\{X^n_{f,i,j}\}$ for path k satisfying Eq. (4)-(6)
      end
    end
    $\lambda^n_f = \max_k \lambda^n_{f,k}$
  end
  Find $\lambda^n = \min_{f \in F^n} \lambda^n_f$
end
$\lambda^* = \sum_{n \in N} \lambda^n$

dynamic flow allocation in **Algorithm 5**, yielding a completed solution for jointly optimized virtualization and routing decision. In particular, network hypervisor is first executed to partition the SDN infrastructure into several subnets for multi-tenants. Once the network slicing is finished, dynamic flow allocation is applied to provide the optimal network throughput and the corresponding flow assignments. Normally, dynamic flow allocation can resolve all impacts from time-varying QoS requirements, network topologies, and link capacities by re-running the allocation accordingly, and network hypervisor only needs to run once and for all. However, if the allocation cannot provide a feasible solution, i.e., λ^* remains zero, QVR will feedback to network hypervisor and enable a better network slicing as well as the achievable flow allocation. Therefore, upon this stage, we have successfully presented an adaptive solution that completely solve the joint virtualization and routing problem in a timely manner, facilitating the practical implementations of multi-tenants' applications in SDNs.

Algorithm 6: QoS-aware Virtualization-Enabled Routing (QVR)

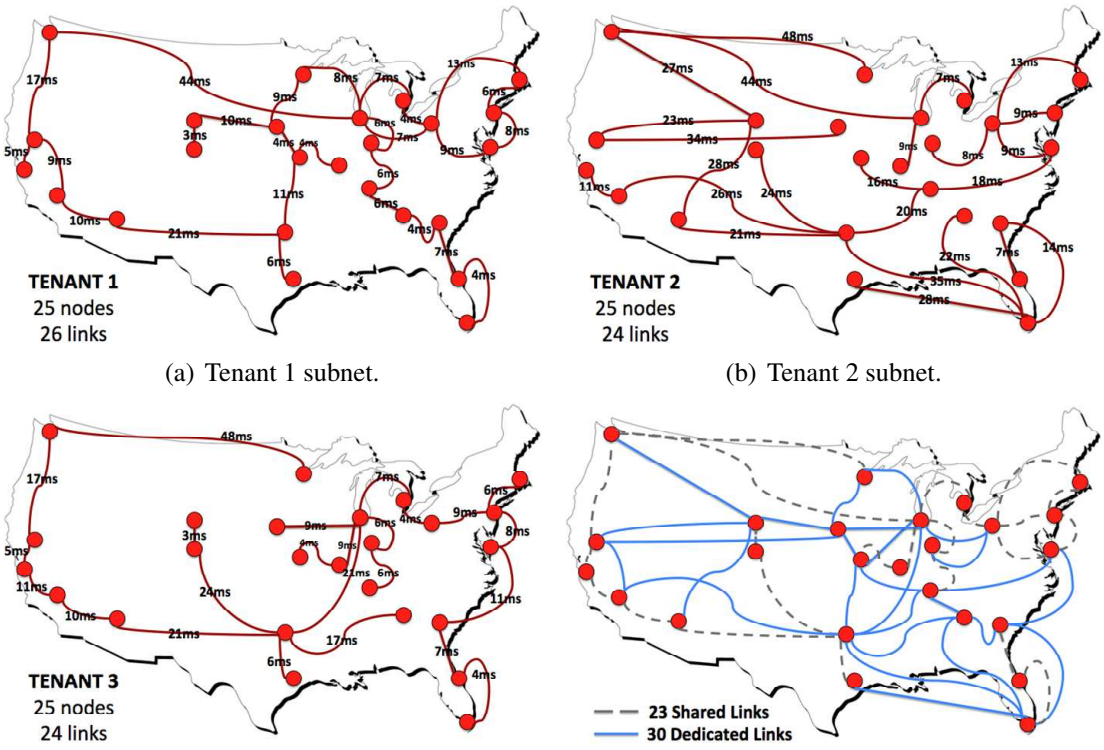
Input : G; $\{w_n^{max}\}$; $\{c_{ij}\}$; $\{D_{n,f}^{max}, J_{n,f}, p_{n,f}^{max}\}$
Output: $\{G_n\}$; λ^*; $\{X_{f,i,j}^n\}$

while $\lambda^* == 0$ **do**
  Network Virtualization Phase
  $\{G_n\} \leftarrow$**Algorithm 1**$(G, \{w_n^{max}\})$
  Flow Allocation Phase
  $(\lambda^*; \{X_{f,i,j}^n\}) \leftarrow$**Algorithm 2**$(\{G_n\}, \cdot)$
end

4.3.5 Performance Evaluation

To evaluate the proposed QVR algorithm in **Algorithm 6**, a Sprint network topology [5] is considered as the SDN infrastructure. It is assumed that there are three tenants, operating in the infrastructure. In particular, tenant 1 generates traffic from real-time applications, which is modeled with Pareto arrivals due to traffic burstiness; tenant 2 and tenant 3 generate traffic from non-real-time applications, which are modeled with exponential arrivals. A random variable $X \in \mathcal{PAR}(\alpha, x_m)$ if it follows Pareto distribution with parameters α and x_m, i.e., $P(X > x) = (x_m/x)^\alpha$. A random variable $X \in EXP(\lambda)$ if it follows exponential distribution with parameter λ, i.e., $P(X > x) = e^{-\lambda x}$. Towards this, we have $\mathcal{PAR}(1.5, 10)$ for tenant 1, $EXP(0.2)$ for tenant 2, and $EXP(0.4)$ for tenant 3. The link service time follows an exponential distribution. Moreover, the highest allowable path delay is selected as the maximum path weight, and the values for three tenants are set as 100ms, 400ms, and 500ms, respectively, according to their specific applications. In the following, we first evaluate the network slicing capability of QVR, and then examine the QVR performance.

Network Slicing: Figure 24 shows the tenant isolation by QVR algorithm, particular from network hypervisor. Specifically, Figure 24(a)-24(c) provide the subnets for tenant 1-3, respectively, and Figure 24(d) further provides the superposition of three tenants, where the shared links are highlighted by the dotted lines. The results imply that after executing the network hypervisor, three subnets are obtained from the original physical network, and the subnets are built to have the minimum shared links. Regarding these few shared links,



(a) Tenant 1 subnet.

(b) Tenant 2 subnet.

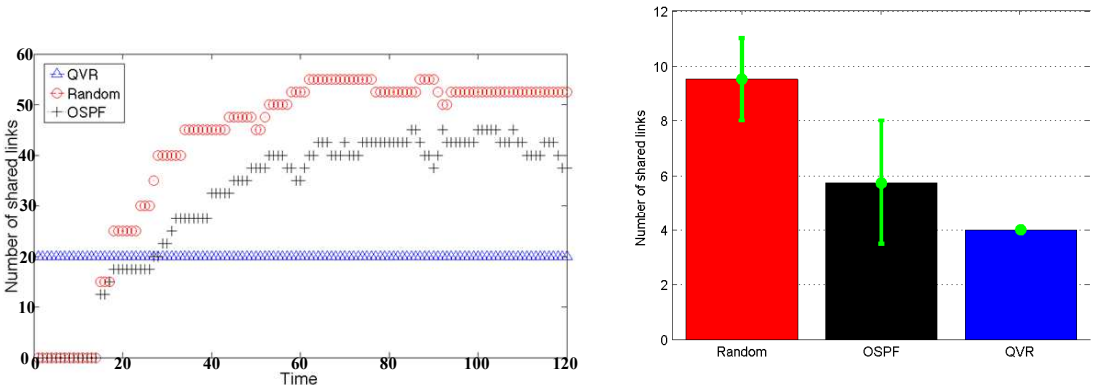
(c) Tenant 3 subnet.

(d) Shared and dedicated links.

Figure 24. Tenants' subnets and shared links.

the slicing of link capacity among sharing subnets will be taken care by the dynamic flow allocation, fulfilling fine-grained network virtualization. Given the subnets of three tenants in Figure 24, in the following we compare the performance of QVR upon this network slicing with the conventional flow allocation solutions.

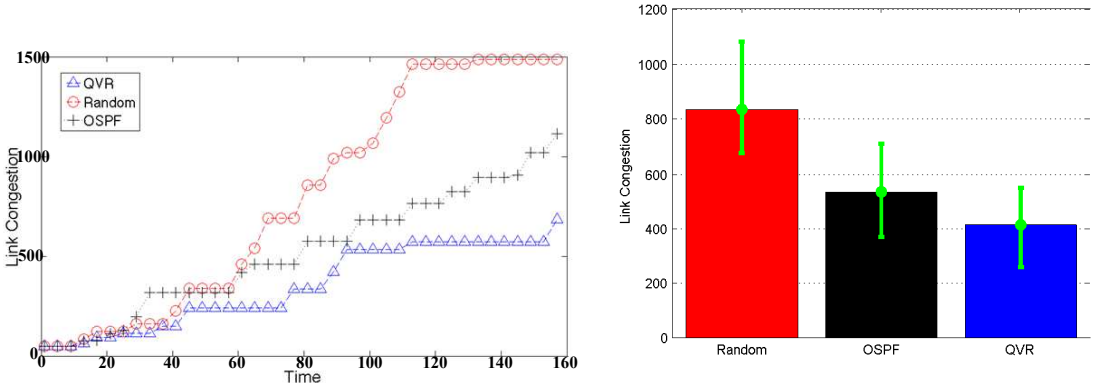
Performance of QVR: the number of shared links and the latency from congested links are first examined through the perspectives of time evolution and multiple experiments for three approaches: OSPF [61], random flow allocation, and QVR. In particular, OSPF decides the flow allocation and corresponding network slicing through shortest-path algorithm, whereas random flow allocation determines those through random path selections. Figure 25 shows the results of shared links, and indicates that under both OSPF and random approach, the link number reaches the maximum value in an early stage. However, QVR can maintain the number of shared link as a constant value due to its network hypervisor, thus achieving better tenant isolation. Moreover, Figure 26 shows the congestion



(a) Time evolution of a single evaluation.

(b) Results of 20 experiments.

Figure 25. Number of shared links with respect to different flow allocations.



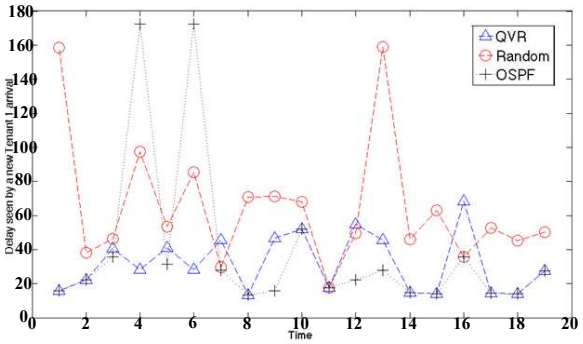
(a) Time evolution of a single evaluation.

(b) Results of 20 experiments.

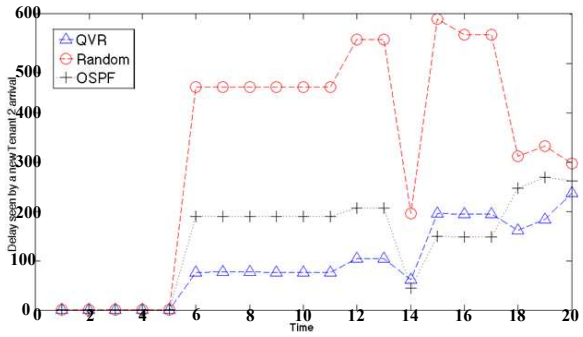
Figure 26. Number of congested links with respect to different flow allocations.

latency, and indicates that QVR performs much better with less latency than OSPF and random solution. The reason for QVR's superiority comes from the optimized route selections that wisely utilizes the link capacity for current flows and thus can provide more bandwidths upon bottleneck links for future flow arrivals. In short, the above results show that QVR accomplishes tenant isolation while improves the congestion latency, allowing more incoming flows from tenants.

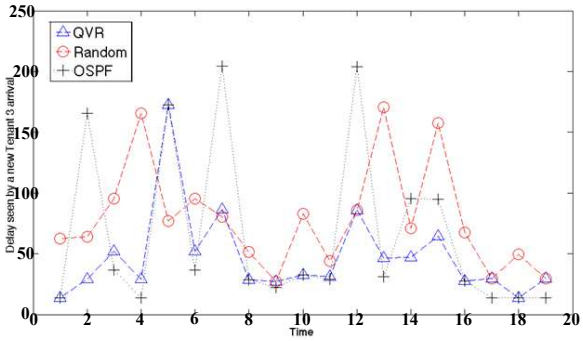
The delay perceived by new flow arrivals of three tenants are provided in Figure 27. In particular, tenant 1 considers real-time applications that brings more bursty traffic and requires much less delay than the non-real-time traffic of tenant 2 and tenant 3. The results



(a) Tenant 1 subnet.



(b) Tenant 2 subnet.



(c) Tenant 3 subnet.

Figure 27. End-to-end packet delay with respect to three tenants.

show that OSPF and random solution cannot fulfill tenant 1's requirements due to the large and highly fluctuated delay, and provides much greater delay for both tenant 2 and tenant 3. On the other hand, QVR provides little delay with less fluctuation for all three tenants, thus confirming its efficacy. In short, QVR meets the need of joint virtualization and routing, facilitating reliable and efficient transmissions for multi-tenants' applications.

4.3.6 Highlights

Network virtualization capacity is essential to support IaaS, thus enabling a wide range of emerging applications. In this section, a joint design of optimized QoS-aware virtualization and routing is addressed as a multi-objective optimization problem. This complex optimization is completely solve in a timely manner through the proposed management tool of QVR in SDN controller. Specifically, with the aid of incorporation between network virtualization and flow allocation, QVR enables an adaptive solution with respect to time-varying QoS requirements, network topologies, and traffic statistics that slices the network resource among multi-tenants' applications and decides the optimal flow routes to fulfill the QoS guarantees for applications. Performance evaluation confirms QVR outperforms the conventional approaches with less shared edges, congestion latency, and traffic delay for multi-tenants. We have presented a novel paradigm to facilitate a virtualization-enabled traffic engineering for centralized controller in practical SDN implementations.

4.4 Traffic Classifier

The last enabling tool is a QoS-aware traffic classifier. Traffic classification is the task of associating network traffic flows with the application that generated it, or of categorizing network traffic flows into different QoS classes (e.g., interactive, bulk data transfer, streaming, and best effort). It is an essential function to enable differentiated resource provisioning and service-based pricing in future broadband mobile systems.

4.4.1 Motivation and Related Work

As SDN manages the network traffic on the basis of "flows", the accuracy and efficiency of the traffic classification (TC) engine plays a crucial role in SDN. Different from the most of the existing work which focuses on identifying the applications that generate the traffic flows, we aim to propose a novel QoS-aware TC framework capable of identifying the QoS class, such as interactive video gaming or bulky data transfer, for different traffic flows in a real-time and cost-efficient fashion. On one hand, providing desired QoS for different

traffic flows is an essential part of the traffic engineering in SDN. Thus, the TC engine has to identify the QoS class for the particular traffic flows so that the suitable routing paths can be chosen. On the other hand, the conventional traffic classification solutions, which aim to identify the exact application of every traffic flow, is not an effective way to identify the QoS class of the traffic flows because many different applications may belong to the same QoS class, which demands the similar QoS requirements. Moreover, as many new applications appear every day, it is time-consuming and impractical to maintain the real-time update of the list of all applications existing within the Internet.

To counter the above-mentioned challenge, we jointly apply machine learning with DPI, in a novel framework that can be fully implemented in a SDN controller. The proposed framework consists of two components: (i) the local traffic identification component at SD switches at the network edge and (ii) the global traffic classifier at the network controller. The former one aims to detect the long-lived, i.e., "elephant" flows among the new incoming ones, while the latter part performs the QoS-aware traffic classification for identifying the QoS class of the traffic flow through a mapping function. The mapping function is simply a function that takes a few features of the traffic flow, e.g., the average packet interarrival time, Hurst parameter and port number, as the inputs and gives the QoS class of the traffic flow as the output. The global traffic classifier at the controller is responsible for learning, building and refining the mapping function based on the historical traffic information. The proposed traffic classification system has three advantages. First, the SD-switches are kept as simple as possible by only incorporating light-weight elephant flow identification module. Second, the network controller is utilized to guarantee the accuracy and the adaptability of the QoS traffic classifier. This is achieved by exploiting the global view of the network flows to build the accurate mapping functions through time-consuming but accurate DPI along with the semi-supervised machine learning that only requires limited information, e.g., first 20 packets, from the elephant flows detected at the SD switches. Third, the whole framework follows a modular design principle so that every component

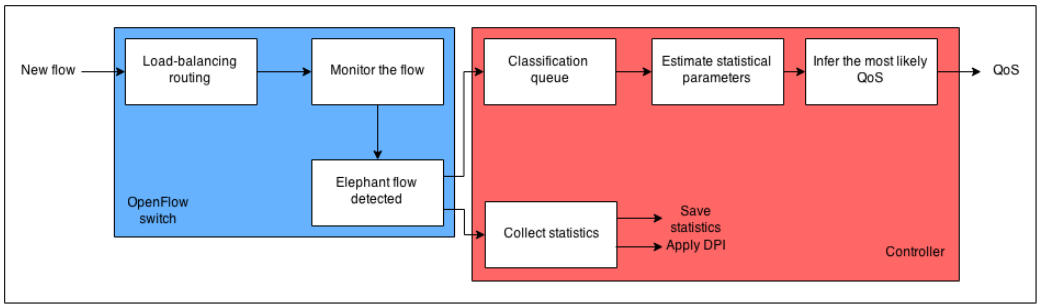


Figure 28. Traffic classification framework scheme.

in the framework can be improved at any time.

4.4.2 QoS-aware Traffic Classification Framework

To conform the SDN architecture, our TC engine is located within the centralized SDN controller. The purposed TC engine performs: (1) efficient network monitoring with low-overhead and minimal switch changes; (2) detection of QoS-significant (i.e., "elephant") flows; (3) QoS-aware traffic classification, and (4) enables services such as application detection using DPI ruining in the network controller. As shown in Figure 28, the system consists of two main parts. The first component is responsible of detecting the QoS-significant flows in the new incoming flows. The second component performs the QoS-aware traffic classification and the related network management tasks. The main process of the operation of the system is explained as follow:

Elephant Flow Detection: after the network flows go through the edge switches, those switches detect the QoS-significant elephant flows. Several approaches have been proposed to detect elephant flows in literature. Since the SDN controller can monitor the network, the detection criterion used is that if a flow uses more that K% of the link bandwidth, it is recognized as an elephant flow, where K could go from 1% to 10% depending on the bandwidth of that link. The OpenFlow defaultly supported pull-based statistics can be utilized in this detection process. By receiving the traffic statistics sent from the switch, the controller gets to know the existence of "elephant" flows.

Statistic Collection and Feature Extraction: in network controller, the ML algorithms

build a mapping function $g(\vec{\mathbf{x}}) = y$ where $\vec{\mathbf{x}}$ corresponds to measurable statistical properties of an elephant flow while y refers to the most likely QoS that flow needs. Therefore, before classifying a flow, two steps need to be done. One is to obtain the measured vector $\vec{\mathbf{x}}$, and the other is to train the ML-based classifier. As long as an elephant flow is detected by a SDN switch, the flow information needs to be sent to the controller through the least congested path immediately. Once the controller gets the flow information, it starts to calculate the statistical properties and group them in $\vec{\mathbf{x}}$. Information is firstly gathered from the packet trace of the flow and then the statistical features are extracted out from that information. The features used to train the ML algorithm and to classify the flows are categorized into the following classes:

- Time information: inter-arrival period;
- Packet information: packet length and direction of the packet;
- Protocol information: IP/Port of source/destination and transport protocol;
- stochastic information: Hurst parameter

To build a real-time classifier only the information of first N ($N = 20$ in our work) packets in a flow is used to calculate $\vec{\mathbf{x}}$. The simulation results proof that using only 20 packets is enough to have a good classification. As the socket information is always the same, the information of the packet belonging to the same socket should be gathered only once.

QoS-aware Traffic Classification: the QoS classifier exploits the semi-supervised ML algorithm, e.g., Laplacian SVM which is hosted inside the centralized SDN controller. Before performing the actual traffic classification, the classifier needs to be trained by following the steps below:

1. Setting up a database storing network traffic traces;
2. Filtering out the ”elephant” flows from the database;

3. Applying DPI to find out the application of each remained ”elephant” flows (i.e., labeling the flows). Note that a significant portion of all flows are still unlabeled due to limited information;

4. Defining QoS classes based on the applications found. Delay, jitter, and loss rate are mainly concerned factors. The corresponding detected applications are assigned to each class as its representative applications. For instance,

   - Voice: GoogleVoice
   - Video conference: Skype, GoogleTalk
   - Streaming: USstream, Sopcast
   - Bulk data transfer: FTP, Mega
   - Interactive data: SSH, Telnet
   - Best-effort traffic: default class

   Notice that not all kinds of application are considered in the QoS class definition, because the only elephant flows enters our TC engine ;

5. Labelling all flows with their corresponding QoS classes to complete the ”labeling” process;

6. Measuring the statistical parameters of each flow and calculating the feature vector $\vec{\mathbf{x}}$ used in the ML algorithm;

7. Training the classifier using semi-supervised learning. In particular, Laplacian SVM is adopted.

Instead of performing fine-grained application classification, coarse-grained classification with better generalization properties is used here. It is assumed that applications that require the same QoS, tend to exhibit similar statistical properties. This is a typical semi-supervised learning assumption [62], called the cluster assumption.

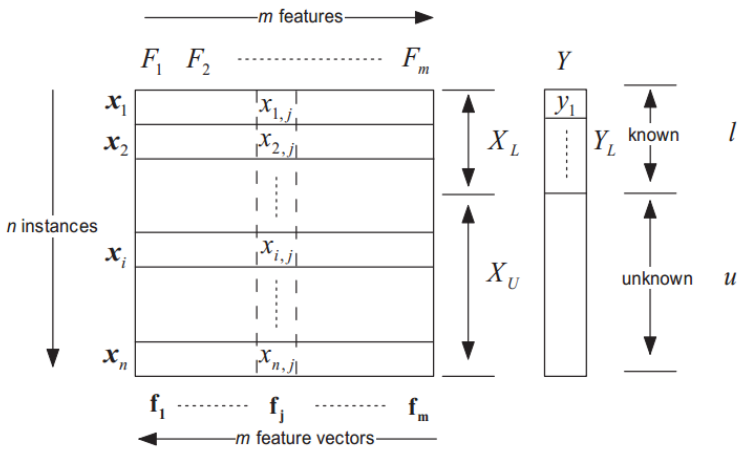


Figure 29. Data structure for semi-supervised learning.

Semi-supervised ML algorithms use labeled, $\mathcal{X}_L$, and unlabeled, $\mathcal{X}_U$, data to infer the classification model. In semi-supervised learning, the data set consists of n vectors $\mathcal{X} = (x_i)_{i\in[n]}$ and two subsets depending on the label. The first one $\mathcal{X}_L = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l)$ are labeled data set with the labels $\mathcal{Y}_L = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_l)$. The second one $\mathcal{X}_U = (\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, ..., \mathbf{x}_{l+u})$ are unlabeled ones. Considering $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_U$ and $l+u = n$, on one hand, when $l = 0$ we do not have labeled data and $\mathcal{X}$ is used for unsupervised learning. On the other hand, when $u = 0$ all samples are labeled and $\mathcal{X}$ is used for supervised learning. Each element $\mathbf{x}_i$ is composed of a series of features. Specifically, each $\mathbf{x}_i$ has m features $\mathbf{f}_i$ so that $\mathbf{x}_i = (\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_m)$. Figure 29 summarizes the structure of the data.

In our scheme, the graph-based semi-supervised learning [63] is adopted which utilizes a graph representation of the data, with each node corresponding to a labelled or unlabelled sample. The graph may be constructed using domain knowledge or similarity of examples, while two common methods are used, which connect each data point either to its k nearest neighbours or to nodes within some distance ϵ. The weight W_{ij} of an edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ is typically set to $e^{\frac{-\|x_i - x_j\|^2}{\sigma}}$. Using the principles behind manifold regularization [64] and the regularization formulation of SVM [65], the SVM can be formulated with the manifold

regularaizer as:

$$f = \arg\min_{f \in \mathcal{H}} \left\{ C \sum_{i=1}^{n} max(0, 1 - yf(x)) + \gamma_{\mathcal{H}} \|f\|_{\mathcal{H}}^2 + \gamma_I \|f\|_I^2 \right\} \tag{32}$$

We adopt the similar methods in [66] to solve the The procedure of solving above problem. The detailed procedures are omitted for brevity. The training phase of the ML-based classifier is usually done offline. With the labeled traffic traces obtained through DPI in 3)-5), when a calculated feature vector $\vec{\mathbf{x}}$ is fed into the classifier, the coefficients of the ML algorithm will be adjusted based on the difference (i.e., error) between the temporary outcome and the actual labeled result. After training with a number of the labeled traces, the coefficients are adjusted well enough to stop the training phase (i.e., the temporary outcome and the actual labeled result match well).

Ground Truth Update and Classifier Re-training: it is known that machine learning learns from the big data and builds a mapping function $g(\vec{\mathbf{x}}) = y$. Thus, having good and meaningful data traces as references is important to build classifiers with good accuracy and generality. However, different types of applications and traffic patterns arise depending on the specific purpose of a network; for example, the applications used in residential networks are different from those used in research networks or data enterprise networks. Furthermore, new applications keep emerging every day and even the current ones may be experiencing updates which change their functionalities and consequently change the statistical properties. Therefore, it is necessary to deploy a policy to gather new data from the network and update the ground truth database periodically, so that it can be used to re-train the QoS classifier after a duration of t_{update}. Basically, it is required to save N (e.g., $N = 20$) packets of various elephant flows flowing through the network so that later we can update the classifiers used. For real-time requirement, it is desirable that a small number of packets is needed to estimate the statistical features because the parameter estimation should be fast. The following method is proposed to gather the data needed:

- When an elephant flow is detected, its information will be stored with a probability

flow_id	start_time	end_time	local_ip
remote_ip	local_port	remote_ port	transport_p rotocol
operating_ system	process_ name	urls	content_ types

Figure 30. Data structure used for flows in the .info file.

of p, the value of which depends on the network state;

- If an elephant flow is selected to be stored, then one of the intermediate SDN switch along the path of the flow will be randomly selected and send the information (i.e., N packets) of the flow to the controller. Those packets received by the controller will be stored in a historical database.

4.4.3 Performance Evaluation

We conducted traffic classification simulations on the real internet data, which was captured by the Broadband Communication Research Group in UPC, Barcelona, Spain. The data set is a 59GB traffic trace file in which the packets of the internet traffic flows on the campus were stored. Note that, the payload of most packets are not stored for the consideration of storage space and privacy issues. Before starting the traffic classification engine, several preprocessing stages need to be done in order to trim the traffic trace file into a set of feature vectors of the traffic samples, which are fed into the classifier.

First, for the data set in use, the flow information was also collected during its capturing process. The packets in the data set has already been categorized into more than 760000 traffic flows. The flow information is stored in a .info file with a format shown in Figure 30. After going through all the flows, we found that more than half of the flows were torrent traffic, so that from the perspective of building a more diverse data set, 440000 torrent flows were removed which finally produced the 59GB .csv file. Second, based on the design of

QoS Class	Applications
Voice/Video Conference	Skype, QQ, Google Hangout, ...
Interactive Data	Gaming, Web, Http Services, ...
Streaming	PPStream, Vimeo, SopCast, Putlocker, ...
Bulk Data Transfer	FTP, Torrent, Dropbox, ...

Table 2. Example of QoS classes and corresponding applications.

Total Flows		Training Set		Testing Set				
L	U	L	U	L				U
1869	1508	1486	1508	383				0
				C1	C2	C3	C4	
				11	29	141	202	

Table 3. The ingredients of the data set used.

our traffic classification framework, elephant flow detection/filtering was conducted on all the flows in the data set to extract all elephant flows which were the objective flows in our classification simulation. As a result, there are 3377 flows satisfying the requirement.

The third step was to label each flows selected in the previous step. All the flows went through a DPI module which outputs the label of each flow. The labeled flows were then categorized into 4 QoS classes including voice/video conference, interactive data, streaming, and bulk data transfer. Examples of the mapping policy between the application and the QoS class is shown in TABLE 2. Because it is hard for us to put all the existing applications on our list, there exist a large number of unlabeled flows within all the flows. Moreover, as new applications appear every day, it is not possible to have all flows labeled in a real traffic classification application. In the data set used, there were 1508 unlabeled flows among all 3377 flows. And the data set was further cut into two groups, the training set and the testing set as in TABLE 3. The ratio between the size of the training set and the size of the testing set is 7.82:1. Note that, to correctly run the testing process, all the flows in the testing set should be labeled, becasue the unlabeled flows cannot be verfied with an unknown application after going through the classifier.

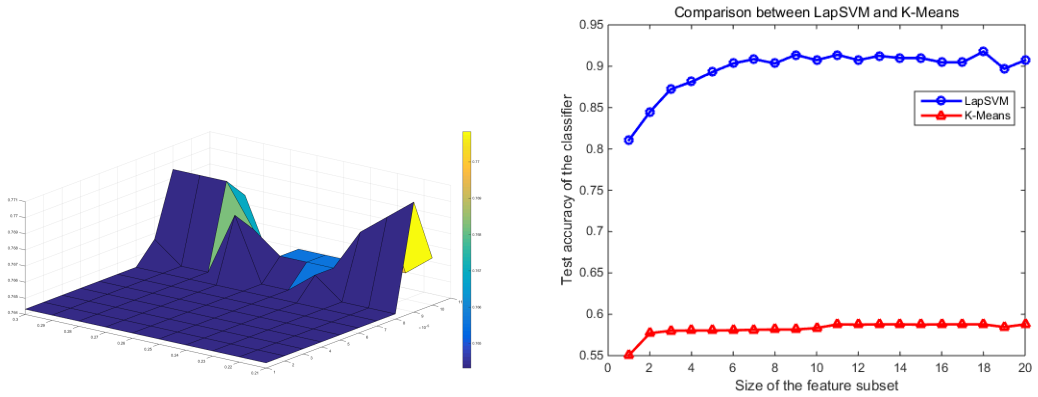
The fourth step was extracting features from each flow. In the evaluation process as

Feature	Explaination
HPktLenSD	Entropy of the packet length from Src. to Dst.
dstPort	Dst. port
srcPort	Src. port
HPktLenDS	Entropy of the packet length from Dst. to Src.
avgPktLenDS	Average length of packets from Dst. to Src.
avgPktLenSD	Average length of packets from Src. to Dst.
rspndPkt	Packets to respond from Src. to Dst.
minPktLenDS	Minimum length of packets from Dst. to Src.
pktIntDegree	Packet interactivity degree from Src. to Dst.
medPktLenSD	Median of the packet length from Src. to Dst.

Table 4. Final subset of features used.

in Figure 31(a), 60 features were extracted from the raw flow data, and as we mentioned before, these 60 features can be categorized into four groups, including time information, packet information, protocol information, and Hurst parameters. However, as a general issue existing in all machine learning based traffic classification research, 60 features are way too many for all being used in the training process of the classifier, because the dimension is too high while the training set is too small as compared with the level of the dimension. This would lead to the severe overfit of the classifier and make the classifier with a bad generalization ability, which means both the accuracy (bias) and the variance would not be good. So that we conducted a feature selection algorithm (i.e., Wrapper) in which the forward selection was employed. Since the Laplacian SVM classifier used has two parameters (i.e.,λ and σ) affect the performance of the classifier a lot, a semi-greedy search on (λ, σ) pairs with two steps (i.e., coarse search and fine search) was used to find the classifier with the best performance. Firstly, in coarse search, the log spaces of the two parameters were used, and the result could provide a small region of (λ, σ) where the accuracy of the classifier is better than elsewhere. Based on our simulation, the area around ($\lambda = 0.00005, \sigma = 0.25$) was expected. Secondly, a fine search among $\lambda = 0.00001 : 0.0001, \sigma = 0.21 : 0.23$ was conducted to search the classifier with the highest accuracy.

Based on the accuracy of the test results on different subset of features, we finally chose



(a) The distribution of 60 features. (b) The accuracy of the classifier with 60 features.

Figure 31. The comparison of traffic classifiers.

a subset of 9 features among all 60 features, considering the complexity of the classification system. The 9 features are listed in TABLE 4. The comparison between the performance of our classifier and the existing K-means algorithm based classifier in terms of testing accuracy is shown in Figure 31(b). Our TC framework using Laplacian SVM as the semi-supervised machine learning algorithm outperforms the previous semi-supervised machine learning scheme using K-means algorithm [67]. In addition, it can bee seen that when the number of features selected into the subset reaches 7-9, the test accuracy exceeds 90% which is an acceptable value in the traffic classification area.

4.4.4 Highlights

In this section, a QoS-aware traffic classification framework for SDN is proposed, where traffic flows are categorized into different QoS classes. The QoS parameters inferred could be used to re-route efficiently elephant flows to meet the resource utilization goals. In particular, semi-supervised machine learning is employed in the QoS classifier to deal with the traffic with unknown applications. Since the feature extraction only uses the first several packets of a flow, the engine runs in a real-time fashion. Also, the traffic classification framework is adaptive to different kinds of networks by periodically re-training the ground truth database and the QoS classifier.

CHAPTER 5

SOFTWARE-DEFINED TRAFFIC ENGINEERING FOR SOFTAIR

In this chapter, we propose new traffic engineering solutions designed to leverage the full potential of the SoftAir architecture. Specifically, BS clustering, throughput-optimal scheduling, and QoS-aware routing solutions proposed in this chapter are directly supported by the enabling tools discussed in Chapter 4. In particular, the distributed traffic classification solutions in Chapter 4.4 provide fine-grained, accurate, and fast QoS classification for incoming traffic. By leveraging this feature, differentiated treatments, e.g., clustering, scheduling, and routing solutions proposed in this chapter, can be provided to different network applications, service providers, and virtual operators. Moreover, the control traffic balancing and optimal network planning algorithms in Chapters 4.1-4.2 make collaborative gain through a large number of RRHs possible, and enable dynamic BS clustering. The delay-based throughput-optimal scheduling algorithms in this chapter can achieve theoretical performance limits, since the largest possible network capacity region is provided by dynamic SD-BS formation. Last but not the least, the network virtualization algorithms in Chapter 4.3 enable adaptive routing algorithms, e.g., QoS-aware routing, that can simultaneously offer the best QoS performance for fundamentally different traffic flows.

5.1 Dynamic Base Station Formation for Solving NLOS Problem in 5G Millimeter-wave Communication Systems

Millimeter-wave communication is one of the enabling technologies to meet high data-rate requirements of 5G wireless systems. Millimeter-wave systems due large available bandwidth enable gigabit-per-second data rates for line-of-sight (LOS) transmissions in short distances. However, for non-line-of-sight (NLOS) transmissions, millimeter-wave systems suffers performance degradation because the received signal strengths at user equipments

(UEs) are not satisfactory. In this section, the NLOS problem in millimeter-wave systems is treated from SoftAir perspective. In particular, a so-called dynamic base station (BS) formation is introduced, which adaptively coordinates BSs and their multiple antennas to always satisfy UEs' QoS requirements in NLOS cases. First, the architecture for software-defined millimeter-wave system is introduced, where RRHs coordination is explained and millimeter-wave channel model between RRHs and UEs is analyzed. A ubiquitous millimeter-wave coverage problem is formulated, which jointly optimizes RRH-UE associations and beamforming weights of RRHs to maximize the UE sum-rate while guaranteeing QoS and system-level constraints. After proving the np-hardness of the coverage optimization problem with non-convex constraints, an iterative algorithm is developed for dynamic BS formation that achieves ubiquitous coverage with high data rates in LOS and NLOS cases. Through successive convex approximations, the proposed dynamic BS formation algorithm transforms the original mixed-integer nonlinear programming into a mixed-integer second-order cone programming, which is efficiently solved by convex tools. Simulations validate the efficacy of our solution that completely solves NLOS problem by facilitating ubiquitous coverage in 5G millimeter-wave systems.

5.1.1 Motivation and Related Work

Millimeter-wave communication at 30-300 [GHz] is one of the enabling technologies to meet high data-rate requirements (10 [Gbps] peak rate and 100 [Mbps] cell-edge rate) of 5G wireless systems [68, 69]. This new-type communication brings much wider transmission bandwidths (500 [MHz] or more per channel as compared with 5-20 [MHz] in current microwave communication), and the small wavelength facilitates large antenna array and antenna technology at BSs. However, experiments [70] show that millimeter-wave communication suffers from several limitations (e.g., short-range distances, inevitable blockage effects, and sparse-scattering radio patterns). At this high band [71], energy consumption dramatically increases due to air absorption and shortens communication distances even for LOS transmissions. Moreover, obstacles (like buildings, vehicles, tree branches, foliage)

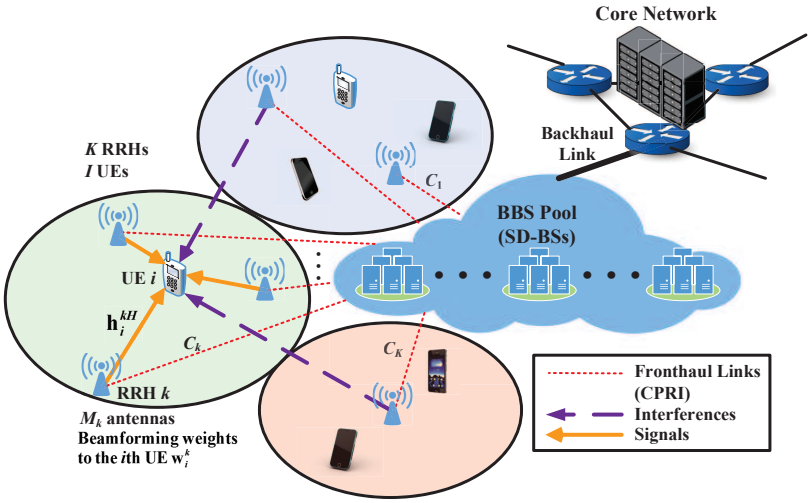


Figure 32. Network architecture of SoftAir [2] for 5G millimeter-wave cellular systems.

may block signals and cause NLOS transmission problems. Also the sparse scattering induces increased channel correlation and narrow beams with less side lobes. These problems jointly impede millimeter-wave transmissions with obstacles and directional beams, causing the NLOS problem.

To overcome the NLOS challenge in millimeter-wave communication, the coordination among multiple BSs and their multiple antennas [72–74], such as coordinated multi-point (CoMP), might serve as a possible solution that enables dynamic coordination between BSs to guarantee good received signal strengths at the UEs. However, in current cellular network architectures, such a coordination of BSs is very limited. In the current architectures, control signaling for BS coordination needs to traverse the access network gateways and costly backhaul links, where the very high latency and limited transmission capacity among BSs can be the reasons for the infeasibility of BS coordination [2].

Our objective is to solve the NLOS problem of millimeter-wave communication by using SoftAir architecture, and consequently facilitate ubiquitous millimeter-wave coverage by the newly introduced dynamic base station formation, which adaptively coordinates BSs and their multi-antennas to always satisfy the QoS requirements of UEs. As shown in Figure 32, SoftAir architecture using low-latency high-bandwidth fronthaul links to realize

accurate, high-resolution synchronization among RRHs and enable flexible RRH coordinations. Moreover, we investigate the millimeter-wave channel model between UEs and RRHs to show the unique characteristics of millimeter-wave transmissions. The channel effects are studied with respect to three link-states: LOS, NLOS, or Outage. Further, we analyze the millimeter-wave coverage problem that jointly optimizes associations between RRHs and UEs as well as the beamforming weights of millimeter-wave RRHs. The objective function is how to maximize the achievable UE sum-rate while guaranteeing UEs' QoS requirements and system-level constraints with respect to (i) RRH-UE associations, (ii) fronthaul link capacity between the baseband server (BBS) pool and RRHs, and (iii) beamforming weights of RRHs.

We prove that the underlying coverage optimization problem with non-convex constraints is np-hard. Thus, we propose an iterative algorithm for dynamic BS formation to obtain optimal solutions in RRH-UE associations and beamforming vectors. By exploiting successive convex approximations [44, 75], we transform the original mixed-integer nonlinear programming (MINLP) of the coverage problem into a mixed-integer second-order cone programming (MISOCP). The final iterative convex programming is efficiently solved by commercial convex tool, i.e., CPLEX [76] or MOSEK [77]. Simulation results confirm that the proposed dynamic BS formation algorithm completely overcomes the NLOS problem, satisfies all UEs' QoS requirements, and outperforms conventional millimeter-wave association and suboptimal beamforming schemes. To the best of our knowledge, this work is the first to propose dynamic millimeter-wave base station formation through software-defined system design, which effectively optimizes UE sum-rate and achieves ubiquitous millimeter-wave communication coverage for 5G wireless systems.

Notations: bold uppercase and lowercase letters denote matrices and vectors, respectively. $\mathbb{C}$ denotes the set of complex numbers. $\mathbb{E}[\cdot]$ and $\Pr[\cdot]$ denote the expectation and probability operator, respectively. x^*, $\Re\mathfrak{e}\{x\}$, $\Im\mathfrak{m}\{x\}$, and $|x|$ respectively represent the complex conjugate, real part, imaginary part, and absolute value of complex variable $x \in \mathbb{C}$. $\mathbf{x}^T$, $\mathbf{x}^H$, and $\|\mathbf{x}\|_2$ represent the transpose, Hermitian, and two-norm of vector $\mathbf{x}$, respectively.

5.1.2 Software-defined Millimeter-wave Communication Systems

In this section, we introduce 5G millimeter-wave communication system, which consists of software-defined cellular systems and millimeter-wave communication.

Software-Defined Cellular System: we consider a multi-user, multi-cell SoftAir [2] downlink system as in Figure 32. Specifically, SoftAir comprises three main parts: (i) the centralized BBS pool, which connects to the core network via backhaul links and consists of software-defined BSs (SD-BSs), (ii) RRHs equipped with antennas, which are remotely controlled by SD-BSs and serve UEs' transmissions, and (iii) low-latency high-bandwidth fronthaul links (fiber or microwave) using common public radio interface (CPRI) for an accurate, high-resolution synchronization among RRHs. Thus, SoftAir can provide accurate channel state information of the entire network to the BBS pool through the flexible design of SD-BSs and RRHs as well as control traffic forwarding technique [35, 78], and consequently enhance significantly the evolvability, scalability, and cooperativeness of distributed RANs. As shown in Figure 32, let $\mathcal{K} = \{1, \ldots, K\}$ and $\mathcal{I} = \{1, \ldots, I\}$ denote the set of RRHs and UEs in the SoftAir system, respectively. We assume that the kth RRH ($k \in \mathcal{K}$) equips with M_k antennas and each UE has a single antenna. All the RRHs are connected to the BBS pool via the fronthaul links, where the kth link between the kth RRH and the pool has a predetermined capacity C_k. Suppose that each UE is served by a specific group of associated RRHs, and a RRH can serve multiple UEs at the same time. To express the association status between RRHs and UEs, we introduce the following binary variables as the indicators. In particular, while RRHs can be active to serve UEs or shutdown to save the energy consumption, $\{a_k, k \in \mathcal{K}\}$ denotes the activity of RRHs as

$$a_k = \begin{cases} 1, & \text{the } k\text{th RRH is in active mode;} \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

Also, $\{b_{ik}, i \in \mathcal{I}, k \in \mathcal{K}\}$ denotes the association between RRHs and UEs as

$$b_{ik} = \begin{cases} 1, & \text{the } i\text{th UE is served by the } k\text{th RRH;} \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

Furthermore, in order to characterize the group (cluster) serving of RRHs, the clustering indicators $\{N_{ik}, i \in \mathcal{I}, k \in \mathcal{K}\}$ are introduced as

$$N_{ik} = \begin{cases} 1, & (i,k) \in \mathcal{L}; \\ 0, & (i,k) \notin \mathcal{L}, \end{cases} \tag{35}$$

where $\mathcal{L} = \{(i,k)|i \in \mathcal{I}, k \in \mathcal{N}_i\}$ denotes the predetermined set of feasible association and $\mathcal{N}_i$ denotes the set of near RRHs for the ith UE, which can be determined based on the distance or channel gain from RRHs to each UE. From these variable definitions, we can obtain the equality $a_k = 1 - \prod_{i=1}^{I}(1 - b_{ik}N_{ik}), \forall k \in \mathcal{K}$ and two sets of association constraints between RRHs and UEs as follows:

$$a_k \geq b_{ik}N_{ik}, \forall i \in \mathcal{I}, k \in \mathcal{K}; \tag{36}$$

$$\sum_{k=1}^{K} b_{ik}N_{ik} \geq 1, \forall i \in \mathcal{I}. \tag{37}$$

Eq. (36) implies that a RRH is in active mode if it is associated with at least one UE. Eq. (37) ensures that each UE is served by at least one RRH.

Millimeter-Wave Communication: in SoftAir downlink system, we introduce the precode vectors (i.e., beamforming weights) at RRHs that realize multi-antenna millimeter-wave transmissions from RRHs to UEs. Let $\mathbf{w}_i^k \in \mathbb{C}^{M_k \times 1}$ be the linear downlink beamforming vector at the kth RRH corresponding to the ith UE, $\mathbf{w}_i \triangleq [\mathbf{w}_i^{1T}, \ldots, \mathbf{w}_i^{KT}]^T \in \mathbb{C}^{M \times 1}$ with $M = \sum_{k \in \mathcal{K}} M_k$ be the set of beamforming vectors to the ith UE, and $\mathbf{W} \triangleq \{\mathbf{w}_1, \ldots, \mathbf{w}_I\} \in \mathbb{C}^{M \times I}$ be the network beamforming design. Let $s_i \in \mathbb{C}$ denote the signal intended for the ith UE with unit power (i.e., $\mathbb{E}[s_i^* s_i] = 1$.) Then, the kth RRH transmits the signal $\mathbf{x}_k = \sum_{i=1}^{I} \mathbf{w}_i^k s_i$ to UEs, and the ith UE receives the signal $y_i \in \mathbb{C}$ as

$$y_i = \sum_{k=1}^{K} \mathbf{h}_i^{kH}\mathbf{x}_k + \eta_i = \mathbf{h}_i^H \mathbf{w}_i s_i + \sum_{j=1, \neq i}^{I} \mathbf{h}_i^H \mathbf{w}_j s_j + \eta_i, \tag{38}$$

where $\mathbf{h}_i \triangleq [\mathbf{h}_i^{1T}, \ldots, \mathbf{h}_i^{KT}]^T \in \mathbb{C}^{M \times 1}$, $\mathbf{h}_i^k \in \mathbb{C}^{M_k \times 1}$ denotes the channel coefficient vector from the kth RRH to the ith UE, and $\eta_i \sim \mathcal{CN}(0, \sigma^2)$ is the zero-mean circularly symmetric Gaussian noise with the noise power σ^2. Different from conventional microwave communication, millimeter-wave transmissions have the following special characteristics: short-range

communication, inevitable blockage effects, and sparse-scattering radio patterns [70, 74]. These jointly affect the downlink channel modeling (more specifically, channel coefficient vectors from RRHs to UEs) and necessitate the rigid analysis of (i) LOS, NLOS transmissions (i.e., blockage) as well as (ii) directional beams (i.e., the directivity). Specifically, we model the channel vector $\mathbf{h}_i^k$ as

$$\mathbf{h}_i^k = \left(l_i^k D_i^k \mathbf{\Phi}_i^k\right)^{1/2} \xi_i^k \in \mathbb{C}^{M_k \times 1} \tag{39}$$

where l_i^k is the large-scale path loss in power (which might also include lognormal shadowing), D_i^k is the directivity gain at the ith UE, $\mathbf{\Phi}_i^k \in \mathbb{C}^{M_k \times M_k}$ is the covariance matrix for antenna correlations in small-scale fading, and $\xi_i^k \sim \mathcal{CN}(\mathbf{0}, \mathbf{I}_{M_k})$ is the fast-fading channel vector.

We further investigate the millimeter-wave channel effects in Eq. (39) with respect to the feasible blockage information. Specifically, if the obstacles (e.g., buildings, vehicles, tree branches, foliage) are well understood in the geographic area, the transmissions between each RRH-UE pair can be categorized into one of the three link-states: LOS, NLOS, or Outage. First, a LOS state occurs when there is no blockage between the RRH and the UE. Assume that in each LOS link, there is no beamforming alignment errors (e.g., the RRH and the UE estimate the angles of arrival and adjust their antenna steering orientations accordingly), and the covariance matrix $\mathbf{\Phi}$ has rank one for all RRH antennas (due to few multi-paths for LOS millimeter-wave channels). This implies that for the LOS link between the kth RRH and the ith UE, the eigenvalue decomposition of the covariance matrix can be modeled as $\mathbf{\Phi}_i^k = M_k \mathbf{u}_i^k \mathbf{u}_i^{kH}$ with a unit vector $\mathbf{u}_i^k \in \mathbb{C}^{M_k \times 1}$. We model the corresponding channel vector as

$$\{\mathbf{h}_i^k; \text{LOS link}\} = \sqrt{M_k l_{iL}^k}\, \mathbf{u}_i^k \tag{40}$$

where l_{iL}^k is the path-loss modeling for a LOS link. Second, a NLOS state occurs when the RRH-UE link is blocked, and the covariance matrix in a NLOS link is similar to the case

for microwave communication. We then model the NLOS channel vector as

$$\{\mathbf{h}_i^k; \text{NLOS link}\} = \left(l_{iN}^k \mathbf{\Phi}_i^k\right)^{1/2} \xi_i^k \tag{41}$$

where l_{iN}^k is the path-loss modeling for a NLOS link. Third, an outage state occurs when no millimeter-wave communication link can be established as the path loss between the RRH and the UE is so high [79]. In practice, this outage implies the case when the path loss in either a LOS or a NLOS state is sufficiently large, and it is a more accurate modeling at millimeter-wave frequency from experimental results [70]. In particular, we have the outage channel vector as

$$\{\mathbf{h}_i^k; \text{Outage link}\} = \mathbf{0}^{M_k \times 1}. \tag{42}$$

Finally, we formulate the path loss with respect to these three states for the link between the kth RRH and the ith UE as

$$l_{iL}^k = (\alpha_L d_i^k)^{-\beta_L};\ l_{iN}^k = (\alpha_N d_i^k)^{-\beta_N};\ l_{iO}^k = 0, \tag{43}$$

where d_i^k denotes the RRH-UE distance, α_L (α_N) can be interpreted as the path loss of the LOS (NLOS) link at a 1 [m] distance, and β_L (β_N) denotes the path-loss exponent of the LOS (NLOS) link. From experimental results, β_N value (can be up to 4) is normally much higher than β_L value (i.e., 2). The parameter values used in the path loss and the occurred probability of three-state modeling can be found in [80, TAIBLE I].

5.1.3 Problem Formulation for Ubiquitous Millimeter-wave Coverage

In this section, our objective is to efficiently realize ubiquitous millimeter-wave coverage of RRHs that supports satisfactory received signal strengths to all geo-distributed UEs. Specifically, we jointly optimize associations betweens RRHs and UEs and beamforming weights of RRHs so that the UE sum-rate is maximized, and UEs' QoS requirements and system-level constraints are satisfied simultaneously. This implies that the degradation of signal strengths due to NLOS transmissions can be completely solved by the solutions

of the coverage optimization problem, particularly when the dense RRH deployment is considered. In the following, we first investigate the constraints with respect to (i) QoS requirements, (ii) beamforming weights, and (iii) fronthaul capacity. Combining these with the association constraints, we then formulate the ubiquitous millimeter-wave coverage problem.

UEs' QoS Requirements: based on the channel modeling of millimeter-wave communication, we formulate the QoS requirements of UEs according to the associated SINR derivations. Let $\bar{\gamma}_i(\mathbf{W})$ and γ_i^{min} denote the received SINR and the minimum SINR requirement of the ith UE, respectively. Following the study in [81, Theorem 1], the SINR constraints of UEs can be formulated as $\forall i \in \mathcal{I}$,

$$\bar{\gamma}_i(\mathbf{W}) = \frac{|\mathbb{E}[\mathbf{h}_i^H \mathbf{w}_i]|^2}{\sigma^2 + \text{var}[\mathbf{h}_i^H \mathbf{w}_i] + \sum_{j=1,\neq i}^{I} \mathbb{E}[|\mathbf{h}_i^H \mathbf{w}_j|^2]} \geq \gamma_i^{min}. \tag{44}$$

Moreover, when only LOS transmissions (without shadowing) are concerned for the ith UE, the corresponding deterministic channel effects can further simplify the SINR of the UE as

$$\gamma_i(\mathbf{W}) = \frac{|\mathbf{h}_i^H \mathbf{w}_i|^2}{\sum_{j=1,\neq i}^{I} |\mathbf{h}_i^H \mathbf{w}_j|^2 + \sigma^2}. \tag{45}$$

Beamforming Weights of RRHs: given $\{s_i, \forall i \in \mathcal{I}\}$ and $\mathbf{w}_i^k$ as UEs' signals with unit power and the precoding vector at the kth RRH for the ith UE's signal, respectively, the transmit power used by this RRH to server the UE is $\mathbf{w}_i^{kH}\mathbf{w}_i^k$. Let P_k^{max} denote the maximum power of the kth RRH. We impose the constraints on beamforming weights of RRHs as follows:

$$\sum_{i=1}^{I} \mathbf{w}_i^{kH}\mathbf{w}_i^k \leq a_k P_k^{max}, \forall k \in \mathcal{K}; \tag{46}$$

$$\mathbf{w}_i^{kH}\mathbf{w}_i^k \leq b_{ik} N_{ik} P_k^{max}, \forall i \in \mathcal{I}, k \in \mathcal{K}, \tag{47}$$

where Eq. (46) limits the total transmit power of RRHs and Eq. (47) ensures that the transmit power from the kth RRH to the ith UE is set to zero if there is no association between them. Furthermore, as the predetermined set $\mathcal{L}$ shows the feasible associations

between RRHs and UEs in Eq. (35), the complexity of computing precoding vectors can be significantly reduced by imposing additional transmit power constraints with respect to RRH-UE association [82]. Specifically, by only allowing RRH-UE links in $\mathcal{L}$, we set the beamforming weights of millimeter-wave communication links as

$$\mathbf{w}_i^{kH}\mathbf{w}_i^k = 0 \text{ if } N_{ik} = 0, \forall i \in \mathcal{I}, k \in \mathcal{K}. \tag{48}$$

Note that Eq. (48) reduces all possible RRH-UE links from IK between K RRHs and I UEs to $|\mathcal{L}|$ links (given that $|\mathcal{L}| << IK$), which in turns dramatically shrinks the possible solution sets of precoding vectors for lower computation complexity.

Fronthaul Capacity With RRH-UE Associations: in SoftAir, the BBS pool directly forwards the compressed precoding vectors and UEs' data streams to the corresponding RRHs; RRHs then precode the baseband signals and transmit to UEs. Specifically, after the BBS pool optimally determines the beamforming vector $\mathbf{w}_i$ for the ith UE, only nonzero weights in $\mathbf{w}_i$ are actually forwarded to the RRHs through the corresponding fronthaul links. More specifically, $\mathbf{w}_i^{kH}\mathbf{w}_i^k = 0$ implies that the kth RRH does not serve the ith UE, and the data stream of the ith UE is not routed from the BBS pool to the kth RRH (via the kth fronthaul link). Given the received SINR of the ith UE $\bar{\gamma}_i(\mathbf{W})$ from RRHs in Eq. (44), we can formulate the corresponding ergodic achievable data rate for the UE as

$$R_i(\mathbf{W}) = B(1-\kappa)\log_2\left(1+\bar{\gamma}_i(\mathbf{W})\right), \tag{49}$$

where B denotes the wireless transmission bandwidth and κ accounts for the spectral efficiency loss due to signaling at RRHs. By neglecting the fronthaul capacity consumption for transferring compressed beamforming vector (as compared to major consumption for data streams) and considering the RRH-UE associations in Eqs. (34)-(35), the per-fronthaul capacity constraints are formulated as

$$\sum_{i=1}^{I} b_{ik}N_{ik}R_i(\mathbf{W}) \leq C_k, \forall k \in \mathcal{K}. \tag{50}$$

This indicates that the total data rate transmitted at the kth RRH should be less than or equal to the rate forwarded by the kth fronthaul link.

Optimization Problem of the Millimeter-Wave Coverage: so far, we have successfully characterized the QoS and system-level constraints for the millimeter-wave coverage problem. To further realize a spectral-efficient coverage design, we aim to maximize the total achievable data rates at UEs as the objective function of the optimization problem. Specifically, given R_i in Eq. (49) as the ergodic achievable rate of the ith UE, the UE sum-rate is provided as $\sum_{i=1}^{I} R_i$; hence, we define the millimeter-wave coverage problem in software-defined millimeter-wave systems as follows.

Definition 3 ***[Ubiquitous Millimeter-Wave Coverage Problem.]*** *Given a software-defined millimeter-wave system with the RRH set $\mathcal{K}$ and the UE set $\mathcal{I}$, and the precoding matrix* $\mathbf{W}$*, the millimeter-wave coverage optimization problem is*

$$\begin{aligned} &\textbf{\textit{Find:}} && a_k \in \{0,1\}, b_{ik} \in \{0,1\}, \mathbf{w}_i^k, \forall i \in \mathcal{I}, k \in \mathcal{K} \\ &\textbf{\textit{Maximize}} && \textstyle\sum_{i=1}^{I} R_i(\mathbf{W}) \\ &\textbf{\textit{Subject to}} && (36), (37), (44), (46), (47), (48), (50) \end{aligned} \quad . \tag{51}$$

5.1.4 Dynamic Base Station Formation via Successive Convex Approximation

Aiming to solve the millimeter-wave coverage problem, in this section, we propose a dynamic base station (i.e., RRHs) formation that optimally determines the following: (i) the association assignments between RRHs and UEs; (ii) the corresponding beamforming weights of millimeter-wave RRHs. In particular, we first prove that the millimeter-wave coverage problem is np-hard. Next, by showing that this coverage problem belongs to mixed-integer nonlinear programming (MINLP) [44], we propose an iterative algorithm of the dynamic BS formation that yields feasible optimal solutions by exploiting convex approximation technique.

Theorem 3 ([80]) *The millimeter-wave coverage problem in Eq. (51) is np-hard.*

SCA-based Dynamic BS Formation: as shown above the millimeter-wave coverage problem is np-hard, this coverage problem is a MINLP problem. Specifically, since Eq.

(50) is non-convex, the coverage problem in Eq. (51) is classified as a non-convex mixed integer programming. It implies that an optimal solution for this problem is very difficult to compute and that the solution would be of little practical interest even if it is possible to obtain. Hence, in this section, we transform the original coverage problem into a tractable formulation so that advanced optimization tools can be used for satisfactory solutions. In particular, we adopt the successive convex approximation (SCA) method [44,75] to approximate non-convex continuous constraints (i.e., Eq. (50)) with series of second-order cone (SOC) constraints and arrive at a MISOCP problem, which can be solved by commercial tools, such as CPLEX [76] or MOSEK [77].

Successive Convex Approximation (SCA): first of all, given the power constraints in Eqs. (46)-(47), we can easily rewrite these hyperbolic constraints into MISOC forms as

$$\|[\mathbf{w}_1^{kT}, \ldots, \mathbf{w}_I^{kT}, \frac{a_k - P_k^{max}}{2}]^T\|_2 \leq \frac{a_k + P_k^{max}}{2}, \forall k \in \mathcal{K}. \tag{52}$$

$$\|[\mathbf{w}_i^{kT}, \frac{b_{ik}N_{ik} - P_k^{max}}{2}]^T\|_2 \leq \frac{b_{ik}N_{ik} + P_k^{max}}{2}, \forall i \in \mathcal{I}, k \in \mathcal{K}. \tag{53}$$

Next, due to the fact that $\text{var}[\mathbf{h}_i^H\mathbf{w}_i] = \mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_i|^2] - |\mathbb{E}[\mathbf{h}_i^H\mathbf{w}_i]|^2$, the constraints of QoS requirement in Eq. (44) can be transformed as

$$(1 + \frac{1}{\gamma_i^{min}})|\mathbb{E}[\mathbf{h}_i^H\mathbf{w}_i]|^2 \geq \sum_{j=1}^{I} \mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_j|^2] + \sigma^2. \tag{54}$$

Similar to the consideration in [72], these beamforming vectors $\mathbf{w}_i^k, \forall i \in \mathcal{I}, k \in \mathcal{K}$ are phase-invariant, which implies that $\mathbf{w}_i^k$ is feasible for QoS requirements if and only if its rotated version $\mathbf{w}_i^k e^{\sqrt{-1}\theta_i^k}$ is. Combing this with the fact that $\mathbf{w}_i^k$ and $\mathbf{w}_i^k e^{\sqrt{-1}\theta_i^k}$ bring the same energy consumption in the objective function of the coverage problem in Eq. (51), we can rewrite the SINR constraints into the following SOC forms:

$$\sqrt{1 + \tfrac{1}{\gamma_i^{min}}}\Re\mathrm{e}\{\mathbb{E}[\mathbf{h}_i^H\mathbf{w}_i]\} \geq \|[\sqrt{\mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_1|^2]}, \ldots, \sqrt{\mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_I|^2]}, \sigma]\|_2, \forall i \in \mathcal{I}; \tag{55}$$

$$\Im\mathrm{m}\{\mathbb{E}[\mathbf{h}_i^H\mathbf{w}_i]\} = 0, \forall i \in \mathcal{I}. \tag{56}$$

In particular, if the SINR constraints follow the deterministic channel effects (e.g., LOS transmission without considering shadowing) as in Eq. (45), the corresponding SOC forms can also be rewritten as $\sqrt{1+\frac{1}{\gamma_i^{min}}}\Re\mathfrak{e}\{\mathbf{h}_i^H\mathbf{w}_i\} \geq \|[\mathbf{h}_i^H\mathbf{W}, \sigma]\|_2$ and $\Im\mathfrak{m}\{\mathbf{h}_i^H\mathbf{w}_i\} = 0$.

Finally, we employ SCA to approximate the non-convex fronthaul capacity constraint into a more tractable form. Specifically, Eq. (50) can be first transformed as

$$\sum_{i=1}^{I} b_{ik}N_{ik}\log_2\left(1+\bar{\gamma}_i(\mathbf{W})\right)^{B(1-\kappa)} \leq C_k. \tag{57}$$

Then, by introducing a set of new variables $\{\alpha_{ik}, \beta_i, \delta_i; i \in \mathcal{I}, k \in \mathcal{K}\}$, we rewrite the capacity constraints into several related inequalities as follows:

$$\textstyle\sum_{i=1}^{I} \alpha_{ik} \leq C_k, \forall k \in \mathcal{K}; \tag{58a}$$

$$(b_{ik}N_{ik})^2 \leq \alpha_{ik}\beta_i; \tag{58b}$$

$$\log_2\left(1+\delta_i\right)^{B(1-\kappa)} \leq \tfrac{1}{\beta_i}; \tag{58c}$$

$$\tfrac{|\mathbb{E}[\mathbf{h}_i^H\mathbf{w}_i]|^2}{\sigma^2+\mathrm{var}[\mathbf{h}_i^H\mathbf{w}_i]+\sum_{j=1,\neq i}^{I}\mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_j|^2]} \leq \delta_i, \tag{58d}$$

where inequalities in Eq. (58a) are in MISOC forms and Eq. (58b) comes from the fact that $(b_{ik}N_{ik})^2 = b_{ik}N_{ik}$. Following the similar approach in Eqs. (52)-(53), Eq. (58b) can be further rewritten into MISOC constraints as

$$\|[b_{ik}N_{ik}, \frac{\alpha_{ik}-\beta_i}{2}]\|_2 \leq \frac{\alpha_{ik}+\beta_i}{2}, \forall i \in \mathcal{I}, k \in \mathcal{K}. \tag{59}$$

Furthermore, adopting similar procedures in [73], we apply the first-order Taylor series expansion as an iterative convex approximation with respect to Eq. (58c) and Eq. (58d) as follows. Specifically, by transforming Eq. (58c) into $1+\delta_i \leq 2^{\frac{1}{\beta_i B(1-\kappa)}}$ and approximating the exponential function around the $(t+1)$th updated point $\beta_i^{(t)}$, we can rewrite the inequalities in Eq. (58c) as the following series of SOC constraints:

$$1+\delta_i \leq \mathfrak{F}_i^{(t)}(\beta_i), \forall i \in \mathcal{I} \tag{60}$$

where $\mathfrak{F}_i^{(t)}(\beta_i) \triangleq 2^{\frac{1}{\beta_i^{(t)}B(1-\kappa)}} - \frac{2^{\frac{1}{\beta_i^{(t)}B(1-\kappa)}}}{(\beta_i^{(t)})^2 B(1-\kappa)}(\beta_i - \beta_i^{(t)})$. Moreover, by transforming Eq. (58d) into $|\mathbb{E}[\mathbf{h}_i^H\mathbf{w}_i]|^2 \leq (1-\frac{1}{\delta_i+1})(\sigma^2+\sum_{j=1}^{I}\mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_j|^2])$, we approximate these non-convex constraints

with another SOC series as

$$|\mathbb{E}[\mathbf{h}_i^H \mathbf{w}_i]|^2 + \frac{|\mathbb{E}[\mathbf{h}_i^H \mathbf{w}_i]|^2}{\delta_i} \leq \mathfrak{G}_i^{(t)}(\mathbf{W}), \forall i \in \mathcal{I} \tag{61}$$

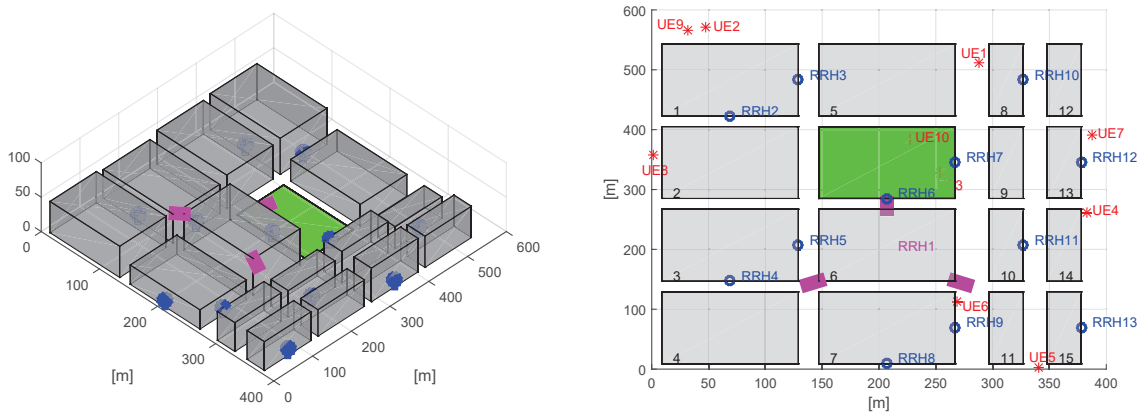
where $\mathfrak{G}_i^{(t)}(\mathbf{W}) \triangleq \sigma^2 + \sum_{j=1}^{I}(2\Re\mathrm{e}\{\mathbb{E}[\mathbf{w}_j^{(t)H}\mathbf{h}_i\mathbf{h}_i^H\mathbf{w}_j]\} - \mathbb{E}[|\mathbf{h}_i^H\mathbf{w}_j^{(t)}|^2])$. In particular, if the deterministic channel effects are considered, the corresponding formulation becomes $\frac{|\mathbf{h}_i^H\mathbf{w}_i|^2}{\delta_i} \leq \sigma^2 + \sum_{j=1,\neq i}^{I}(2\Re\mathrm{e}\{\mathbf{w}_j^{(t)H}\mathbf{h}_i\mathbf{h}_i^H\mathbf{w}_j\} - |\mathbf{h}_i^H\mathbf{w}_j^{(t)}|^2)$. Combining these with the *monotonicity* of logarithmic function and $\delta_i \geq 0$, we can approximate the non-convex coverage problem in Eq. (51) at iteration $t+1$ by the following upper-bounded convex programming with Eqs. (60)-(61) as

$$\begin{aligned}
\textbf{Find:} \quad & a_k, b_{ik}, \mathbf{w}_i^k, \alpha_{ik}, \beta_i, \delta_i, \forall i \in \mathcal{I}, k \in \mathcal{K} \\
\textbf{Maximize} \quad & \textstyle\prod_{i=1}^{I}(1+\delta_i)^{B(1-\kappa)} \\
\textbf{Subject to} \quad & (36), (37), (48), (52), (53), (55), \\
& (56), (58a), (59), (60), (61) \\
& a_k \in \{0,1\}, b_{ik} \in \{0,1\}, \forall i \in \mathcal{I}, k \in \mathcal{K}
\end{aligned} \quad . \tag{62}$$

Note that as the objective and all constraint functions in Eq. (62) follow SOC representation, Eq. (62) becomes a mixed-integer second-order cone programming (MISOCP) problem. This implies that at each iteration, Eq. (62) can be optimally solved through standard MISOCP tools as mentioned. Algorithm 8 summarizes the proposed iterative algorithm.

Algorithm 7: SCA-based Dynamic BS Formation

initialize $\beta^{(0)}$ and $\mathbf{W}^{(0)}$;
set $t = 0$;
repeat
  solve the MISOCP problem in Eq. (62) with $\beta^{(t)}$ and $\mathbf{W}^{(t)}$ for optimal solution $\{\bar{\mathbf{w}}_i^k, \bar{a}_k, \bar{b}_{ik}, \bar{\alpha}_{ik}, \bar{\beta}_i, \bar{\delta}_i\}$;
  set $t := t + 1$;
  update $\beta^{(t)} = \bar{\beta}$ and $\mathbf{W}^{(t)} = \bar{\mathbf{W}}$;
until *convergence or the maximum number of iterations is reached*;



(a) 3D visualization.

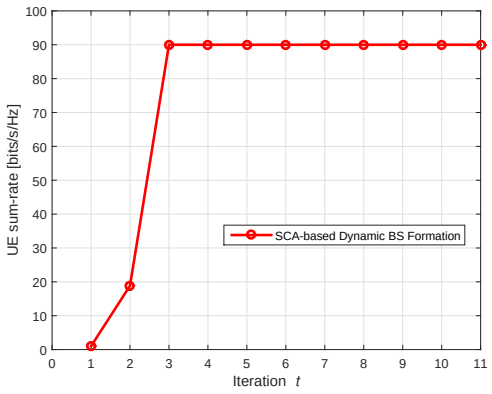
(b) 2D visualization with 10 randomly-distributed UEs.

Figure 33. An urban environmental model of the Madrid grid from METIS [6] with 13 **RRHs deployed.**

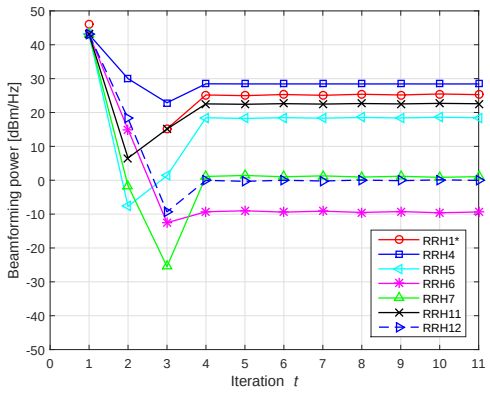
5.1.5 Performance Evaluation

In this section, we present simulation results to evaluate the performance achieved by the dynamic BS formation. Considering practical obstacles in 5G networks, we build software-defined millimeter-wave systems upon METIS [6], which aims to lay the 5G foundation for year 2020 and beyond with realistic consideration of different environments of buildings, roads, park, etc. As shown in Figure 33, the urban environmental model is established, based on observations regarding the city structure of Madrid, and captures more aspects than Manhattan grid. Following the deployment baseline of a three-sector macro station and 12 pico stations for network infrastructure [6], we design 13 RRHs with different specifications: RRH 1, indicated by magenta rectangles, mimics the powerful macro station and has four equipped antennas on the roof of building 6 with maximum transmit power 43 [dBm]; RRHs 2-13, indicated by blue stems to simulate pico stations, has two equipped antennas of 10 [m] height and maximum power 36 [dBm]. We further set the fronthaul capacity as 10 [bits/s/Hz] for each RRH and the carrier frequency as 28 [GHz], and all UEs are randomly-distributed in the street. The three-state path-loss model with lognormal shadowing is considered, and the thermal noise power is set as -101 [dBm/Hz].

Convergence Behavior of Dynamic BS Formation: to illustrate the fast convergence of



(a) Convergence behavior of the UE sum-rate.



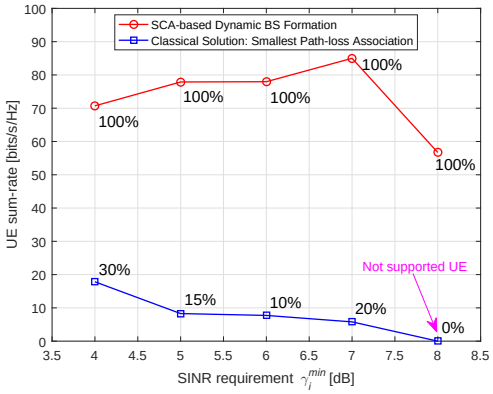
(b) Convergence of the beamforming power at 7 selected RRHs.

Figure 34. Fast convergence of the dynamic BS formation in Algorithm 8.

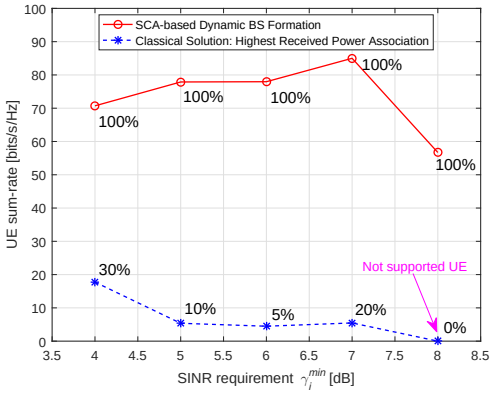
dynamic BS formation in Algorithm 8, Figure 40 shows the convergence behavior of downlink system sum-rate and the beamforming power of 7 selected millimeter-wave RRHs with 10 UEs randomly-distributed, where the required minimum SINR for each UE is γ_i^{min} =6 [dBW]. The results imply that by exploiting SCA, our proposed Algorithm 8 converges very fast after only 4 iterations, which serves as a desired stopping point. Moreover, with the maximum achievable sum-rate around 90 [bits/s/Hz], the average rate per UE can be easily achieved to 4.5 [Gbps], given 500 [MHz] transmission bandwidth from millimeter-wave communication. We demonstrate the efficiency of our proposed BS formation that quickly makes the decision for RRH-UE associations and beamforming weights, facilitating millimeter-wave coverage upon time-varying channels.

Performance Comparison of Dynamic BS Formation and Conventional Millimeter-wave Cell Association: as in [79], two schemes for cell association are commonly studied in conventional millimeter-wave communication.

- *Highest received power association:* Like in microwave communication, the UE-BS association is based on downlink reference signals, which undergo both path-loss and shadowing. Hence, a UE will be served by the BS providing the highest received power to it.



(a) Comparison with the smallest path-loss association.



(b) Comparison with the highest received power association.

Figure 35. UE sum-rates achieved by the proposed dynamic BS formation and two conventional millimeter-wave association schemes, with respect to UEs' SINR requirements. The percentage shows the ratio of supported UEs to total UEs.

- *Smallest path-loss association:* As millimeter-wave communication has less slowly-varying shadowing due to pronounced blockage impact on received signals, UEs might be unable to consider random fluctuations by shadowing. Thus, in this case, a UE will be served by the BS with the smallest path-loss to it.

We evaluate the achievable sum-rates from Algorithm 8 and the above two schemes with respect to UEs' SINR requirements in Figure 41. To realize conventional association schemes, the matched filtering precoding method is exploited as $\mathbf{w}_i^k = \mathbf{h}_i^k / \|\mathbf{h}_i^k\|_2$, the transmit power is equally allocated among the UEs in the same BS's coverage, and the greedy algorithm is used for UE scheduling [83]. Figure 41 shows that the proposed dynamic BS formation significantly outperforms conventional millimeter-wave association with ubiquitous UE coverage and high data rates. This is because our solution accounts the global network conditions by centralized, software-defined system architecture and jointly optimizes the association and precoding decision. The decreasing data-rate trend of our solution is due to fixed-power concern of RRHs. On the other hand, classical solutions rather exploit static and suboptimal designs and thus barely support high UEs' SINR requirements. The results from the smallest path-loss and the highest received power schemes have close

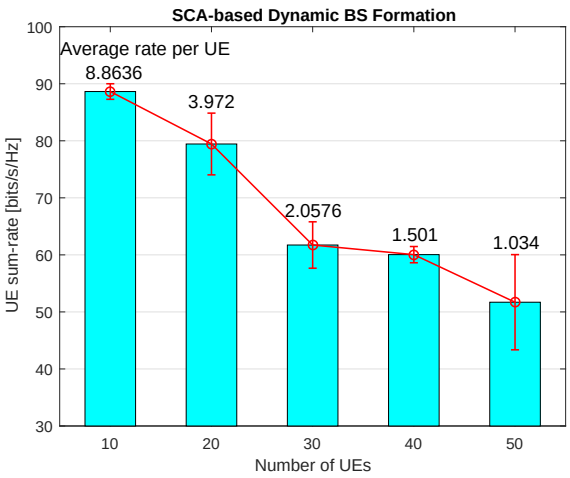


Figure 36. Achievable UE sum-rates by the dynamic BS formation with respect to increasing UEs.

performance. These confirm the efficacy of our solution to solve the NLOS problem and achieves ubiquitous millimeter-wave coverage.

Impact of UE Density: Figure 42 shows the achievable UE sum-rates by Algorithm 8 with respect to the UE density. Specifically, as the number of served UEs increases, both the sum-rates and the average rate per UE decrease, due to lower signal power for the UEs sharing same RRHs' power resources and increasing downlink interferences. Moreover, while the achievable data rate per UE decreases with the increasing UE number, the dynamic BS formation can always support each UE with at least 500 [Mbps] rate through millimeter-wave transmissions, meeting the high data-rate requirements in 5G. The above results validate the adaptiveness of dynamic BS formation, which can be used for the design of 5G millimeter-wave communication.

5.1.6 Highlights

Through W-SDN SoftAir architecture, we develop a dynamic BS formation that achieves a joint optimal design of RRH-UE associations and beamforming weights of millimeter-wave RRHs to facilitate ubiquitous millimeter-wave coverage. The coverage optimization

as a MINLP problem is proved to be np-hard; an iterative algorithm of the dynamic BS formation is proposed through SCA to yield optimal solutions. Simulations show that our solution can always satisfy UEs' QoS requirements with millimeter-wave transmissions and significantly outperforms conventional association schemes with suboptimal beamforming. Our solution can be used to solve the NLOS problem in 5G millimeter-wave systems.

5.2 Delay-based Throughput-optimal Scheduling with Heavy-tailed Traffic for 5G SoftAir: Traffic-awareness

HT traffic (e.g., the Internet and multimedia traffic) fundamentally challenges the validity of classic scheduling algorithms, designed under conventional LT assumptions. To address such a challenge, this section investigates the impact of HT traffic on delay-based maximum weight scheduling (DMWS) algorithms in SoftAir, which have been proven to be throughput-optimal with enhanced delay performance under the LT traffic assumption. First, it is proven that the DMWS policy is not throughput-optimal anymore in the presence of hybrid LT and HT traffic by inducing unbounded queueing delay for LT traffic. Then, to solve the unbounded delay problem, a delay-based maximum power-weight scheduling (DMPWS) policy is proposed that makes scheduling decisions based on queueing delay raised to a certain power. It is shown by the fluid model analysis that DMPWS is throughput-optimal with respect to moment stability by admitting the largest set of traffic rates supportable by the network, while guaranteeing bounded queueing delay for LT traffic. Moreover, a variant of the DMPWS algorithm, namely the IU-DMPWS policy, is proposed, which operates with infrequent queue state updates. It is also shown that compared with DMPWS, the IU-DMPWS policy preserves the throughput optimality with much less signaling overhead, thus expediting its practical implementation.

5.2.1 Motivation and Related Work

Link scheduling is a critical and even most challenging resource allocation functionality in general queueing networks, such as wireless downlinks and uplinks, input-queued

switches, wireless sensor networks, ad-hoc networks, and cloud computing facilities among many others. In all these systems, not all queues can be served simultaneously, due to the constraints from wireless interference or switch matching. To fully utilize the limited network resources, throughput-optimal scheduling policies [84], often referred to as maximum weight (MaxWeight) policies, have been extensively exploited. The throughput-optimal policy can stabilize the network by guaranteeing bounded queueing delay under any feasible loads, without requiring any explicit statistical information of the arriving traffic flows and serving rates. Throughput-optimal scheduling was first introduced in the seminal work [84], which proposes queue length-based MaxWeight scheduling (QMWS), where the flow with the largest queue length is served first. Since then, numerous work has been focused on the variations or extensions of this policy in different settings. However, while these queue length-based policies have been shown to achieve excellent throughput performance, they suffer a substantial queueing delay because the long waiting time of building up large queue lengths [85] is required for a flow to be served eventually. Moreover, the queue length-based MaxWeight scheduling policies even lose the throughput optimality under flow dynamics, where certain flows only have a finite number of packets to transmit and thus cannot bring a sufficiently large queue length to establish desired queueing dynamics [86].

In this section, we aim to analyze the impact of hybrid HT and LT traffic on the throughput optimality of DMWS policies for switch/wireless hypervisors. We propose the delay-based maximum power-weight scheduling (DMPWS). In particular, by jointly exploiting fluid model-based stability and moment analysis, we prove that DMPWS is throughput-optimal with respect to moment stability by admitting the largest set of traffic rates that is supportable by the network, while guaranteeing bounded queueing delay for LT traffic. Intuitively, this feature is achieved by giving higher priorities (i.e., larger power-weight) to LT traffic flows, which provide them sufficient serving opportunities when competing with HT traffic flows. To further enhance the practicability of DMPWS, we propose a

variant DMPWS policy, called infrequent updating-DMPWS (IU-DMPWS), which only needs the infrequent queue-state (i.e., HoL packet delay) measurements. While such infrequent updates of queue information have been exploited for conventional QMWS [87, 88], the impact on the proposed DMPWS is unknown and quite involved to analyze. To this end, we prove that the IU-DMPWS policy still preserves the throughput optimality as its original DMPWS scheme, but is more favored in system implementation, such as uplink scheduling in cellular networks, due to less signaling overhead. There is a clear tradeoff with IU-DMPWS that infrequent updates and lower overhead (or computational cost) come at the expense of larger delay, as verified by simulation results. To the best of our knowledge, this work is the first rigorous analysis for the throughput performance of delay-based scheduling policies with HT traffic. We summarize our main contributions as follows:

- We show that the existing DMWS policy fails to achieve throughput optimality in the presence of HT traffic.
- We prove that the proposed DMPWS policy can achieve the throughput optimality with respect to moment stability, under hybrid HT and LT traffic.
- We further demonstrate that the proposed IU-DMPWS policy preserves the good merits from its original scheme with less signaling cost.

5.2.2 System Model for Switch & Wireless Hypervisors

We consider a multi-queue single-server, time-slotted queueing system, where F queues share a single server. A *traffic flow* $f \in \{1, \ldots, F\}$ is a discrete-time stochastic arrival process $\{A_f(t); t \in \mathbb{Z}_+\}$, which represents the total number of packets that arrive at a queue at the beginning of the time slot t and is i.i.d. from time slot to time slot. The arrival processes or traffic flows are mutually independent. Let $\lambda_f = E[A_f(0)] > 0$ be the rate of traffic flow f and $\lambda = (\lambda_1, \ldots, \lambda_F)$ be the rate vector. Let stochastic processes $\{Q_f(t); t \in \mathbb{Z}_+\}$ and $\{D_f(t); t \in \mathbb{Z}_+\}$ be the number of packets and the queueing delay, respectively, in queue f at the beginning of time slot t. $Q(t) = (Q_1(t), \ldots, Q_F(t))$ captures the queue backlog

at time slot t, and its initial state $Q(0)$ can be an arbitrary element of $\mathbb{Z}_+^F$. A set of flows that can be served simultaneously is called a *feasible schedule*. In our network model, each feasible schedule only contains one flow because only one queue can be served during each time slot. Let S denote the set of all feasible schedules. Then, for each feasible schedule $\pi \in S$, let $\pi_f(t)$ denote the number of packets transmitted from queue f under schedule π at time t. For simplicity, we assume that the service rate is one packet per time slot. Thus, the average service rate of queue f under schedule π is $E[\pi_f(t)] = \pi_f \leq 1$. Accordingly, the time-varying scheduling vector $S(t) = (\pi_1(t), \ldots, \pi_F(t)), \pi \in S$ is determined by the proposed scheduling policy. Let $Y(t) = (Y_1(t), \ldots, Y_F(t))$ denote the idling service at time t. Hence, the set of processes $\{Q(t), D(t), Y(t), S(t)\}$ with $Q(0)$ completely captures the dynamic of an entire stochastic queueing system. Mathematical preliminaries are provided.

Definition 4 (Heavy Tail) *A random variable (r.v.) X is HT, if for all $\theta > 0$, $\lim_{x\to\infty} e^{\theta x} \Pr(X > x) = \infty$, or equivalently, $E[e^{zX}] = \infty$, $\forall z > 0$. A r.v. is LT, if it is not HT, or equivalently, if there exists $z > 0$ so that $E[e^{zA_f(0)}] < \infty$.*

From the existence of the moments, we define the tail index of a nonnegative r.v. X as

$$\kappa(X) := \sup\{k \geq 0 : E[X^k] \leq \infty\}, \tag{63}$$

which defines the maximum order of finite moments that X can have. Moreover, to show the sufficient condition for finite tail indexes [89], we have the following: a nonnegative r.v. X has $\kappa(X)$ if and only if the tail distribution of X satisfies $\lim_{t\to\infty} \frac{\log \Pr(X>t)}{\log t} = -\kappa(X)$.

Definition 5 (Steady-state Stability) *Given the queueing system, if there exists a scheduling policy under which the Markov chain of queue lengths is positive Harris recurrent (i.e., $\{Q(t); t \in \mathbb{Z}_+\}$ converges in distribution), then the queueing network is steady-state stable.*

We further consider moment and strong stability to characterize the delay performance.

Definition 6 (Moment Stability) *Given the single-hop queueing system described above under a specific scheduling policy, if LT traffic flow have bounded average packet delay (i.e., $E[W_f] < \infty, \forall f \in LT$), then the queueing system is moment stable.*

Definition 7 (Strong Stability) *A queueing system is strongly stable if all traffic flows experience bounded average queueing delay (i.e., $E[W_f] < \infty, \forall f \in F$).*

5.2.3 Delay-based Maximum Power-weight Scheduling (DMPWS)

Intuitively speaking, the key problem of MWS is that both LT and HT traffic flows are assigned with the same priority or weight. In this case, the HT traffic flow may receive more services because it has the higher probability to have long queues and high delay, which leads to long waiting time in the queue for the LT traffic. To solve this problem, we propose delay-based maximum power-weight scheduling (DMPWS) [90] that assigns different weights with respect to HT and LT traffic flows. The *power-weight* of a feasible schedule is the sum of the head-of-line (HoL) packet delay $W_f(t)$ up to α_f order, and the DMPWS policy activates a schedule that has maximum power-weight at any given time slot. Specifically, under the DMPWS policy, it implies that the queue f will be served given that

$$W_f^{\alpha_f}(t) = \max_{j \in F} W_j^{\alpha_j}(t), \quad \forall f \in F. \tag{64}$$

The power-weight assigned to each flow should be set to be proportional to the maximum order of the finite moments of the arrival processes $A(t)$ to ensure the network stability. In particular, we set the power-weight of flow $f \in F$ as follows:

$$\alpha_f = \begin{cases} \kappa(A_f(t)) - 1, & \forall f \in HT; \\ c_f > 2, & \forall f \in LT, \end{cases} \tag{65}$$

where $\kappa(\cdot)$ is the tail coefficient defined in Eq. (63) and c_f is an arbitrary constant larger than two. To prove throughput optimality of DMPWS, it is equivalent to show DMPWS can achieve moment stability, i.e., all the LT traffic flows have bounded average queueing delay, as long as the incoming traffic flows are within the network capacity region. With the aid of fluid-limit approximations [91], we first show that DMPWS can achieve steady-state stability by proving that the corresponding fluid model is stable. The key idea is to establish the linear relationship between queueing delay and queue length through fluid

model solutions. With such condition satisfied, we then show that DMPWS can achieve moment stability by exploiting fluid model-based moment analysis under the condition that the incoming traffic flows are within the network capacity region.

Theorem 4 ([91]) *The queueing network is steady-state stable (i.e., positive Harris recurrent) whenever an associated fluid model is stable (i.e., there exist time $T > 0$ such that $q_f(t) = 0$ for all $f \in F$ and $t \geq T$).*

We prove the steady-state stability in Theorem 5 by the Lyapunov technique in fluid domain and by exploiting Theorem 4.

Theorem 5 (Steady-state Stability [90]) *If the incoming traffic rates reside in the network capacity region, the corresponding queueing system is steady-state stable with the DMPWS policy under hybrid HT and LT traffic.*

We further show that the DMPWS policy is throughput-optimal, which ensures that the queue length and queueing delay of all LT traffic flows are of finite mean as long as the traffic rates are within network capacity region.

Theorem 6 (Throughput Optimality [90]) *Consider the DMPWS policy under the hybrid HT and LT traffic. The corresponding queueing system is throughput-optimal with respect to moment stability by ensuring that all LT traffic flows have bounded average queue length and queueing delay (i.e., $E[Q_f] < \infty$ and $E[W_f] < \infty$, $\forall A_f(t) \in LT$).*

Infrequent Queue State Measurements: let the time slots be grouped into *interval* of time T^I. It implies that the $(k+1)$th interval consists of slots $kT^I, \ldots, (k+1)T^I - 1$. Although queue states can change in each slot, these are measured only at the beginning of each interval (i.e., at the beginning of slot kT^I, for $k = 0, 1, \ldots$). Therefore, the interval length T^I denotes the duration between successive sampling instances of the HoL packet delay, and the IU-DMPWS policy follows the DMPWS and only updates the schedule at

each time interval T^I. Specifically, under the IU-DMPWS policy, the scheduling vector $S(t)$ belongs to the set:

$$S(kT^I + l) = \underset{\pi \in S}{\text{argmax}} \{ \sum_{f \in F} W_f^{\alpha_f}(kT^I) \pi_f(t) \}$$

where $k = 0, 1, \ldots$ and $l = 0, 1, \ldots, T^I - 1$. If the set on the right-hand side includes multiple schedules, one of them is chosen uniformly at random. Moreover, the queue dynamic equation is also rewritten as

$$Q((k+1)T^I) = Q(kT^I) + A(kT^I) - T^I[S(kT^I) - Y(kT^I)]. \tag{66}$$

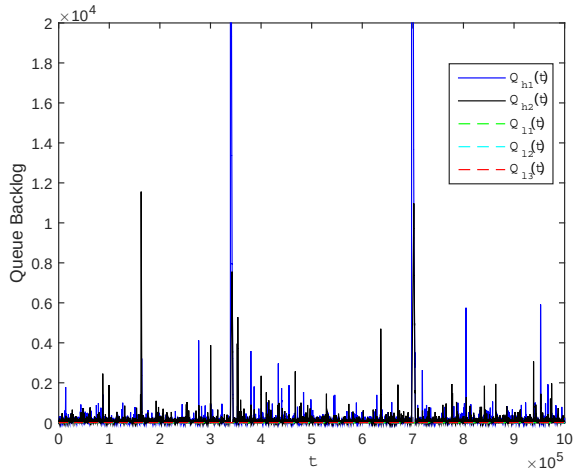
Therefore, given the IU-DMPWS policy, we prove the throughput optimality of IU-DMPWS under the hybrid traffic in the following Theorem 7. It is easy to see that the DMPWS policy is a special case of the IU-DMPWS policy, by considering the case $T^I = 1$.

Theorem 7 ([90]) *The IU-DMPWS policy is throughput-optimal.*

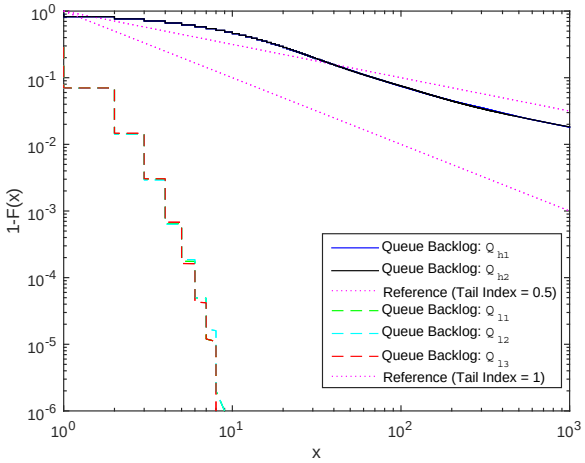
5.2.4 Performance Evaluation

We show that the throughput optimality and bounded delay can be achieved with hybrid traffic by applying the DMPWS policy with Eq. (65). We choose Pareto and Poisson distributions to represent HT and LT distributions, respectively. We assign the queues of HT flows and LT flows with weight 0.5 and 2, respectively. As indicated by Theorem 5 and Theorem 6, under such settings, the DMPWS policy can guarantee that LT traffic flows have bounded average queue lengths and delay, which cannot be achieved by applying the DMWS policy. In particular, Figure 37(a) shows that there is no large queue length for LT flows during the evolution. Figure 37(b) and Figure 37(c) further indicate that for both the tail distributions of queue length and delay of LT flows have a slope or decaying rate greater than one, which implies the queue lengths and packet delay of LT flows have finite mean and thus brings the stable queueing network.

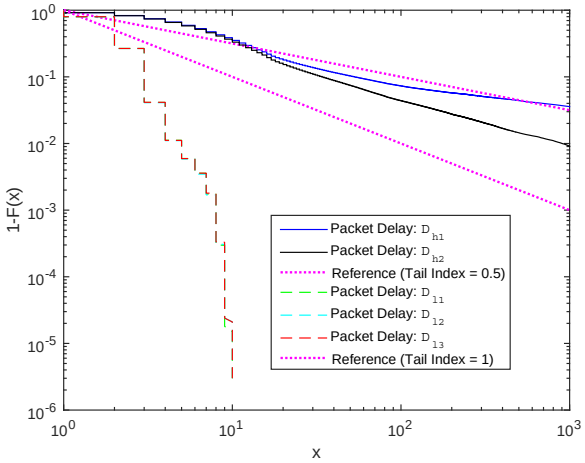
Performance of Infrequent Measurement Update: while we successfully demonstrate the superiority of DMPWS, in the following, we evaluate the impact of infrequent update



(a) Evolution of queue length.

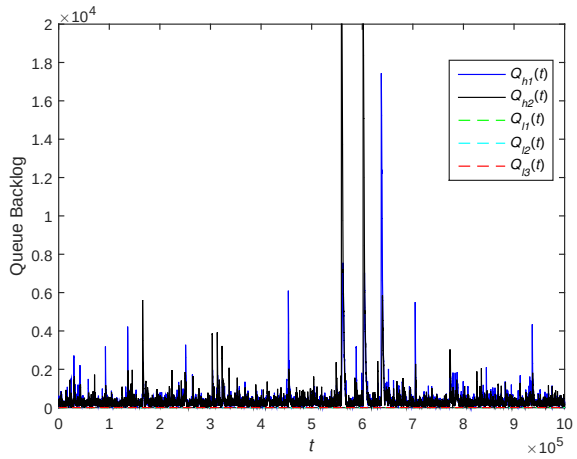


(b) Tail distribution of queue length.

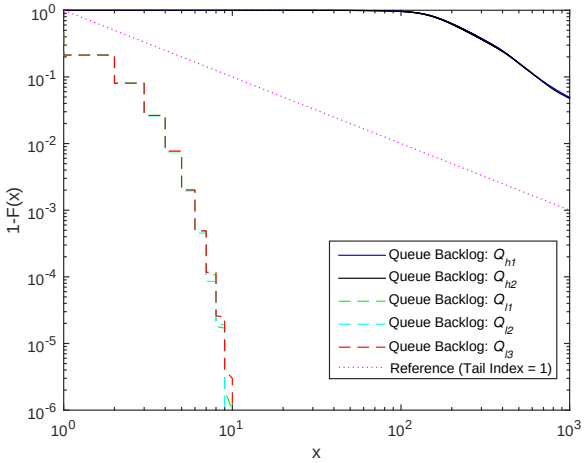


(c) Tail distribution of packet delay.

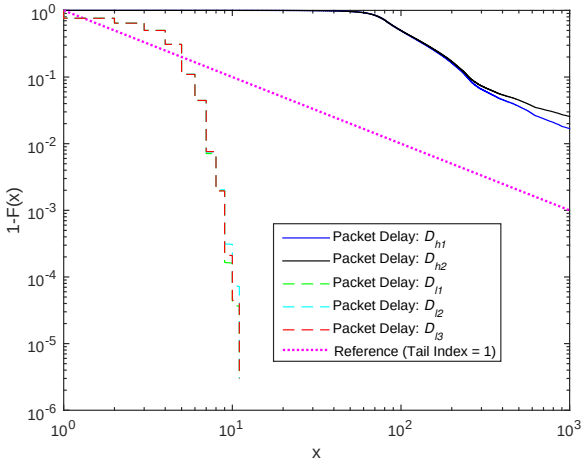
Figure 37. Queue length and packet delay under DMPWS, where the queueing system is stable.



(a) Evolution of queue lengths.



(b) Tail distribution of queue lengths.



(c) Tail distribution of packet delay.

Figure 38. Queue lengths and packet delay under the IU-DMPWS policy with $T^I = 2$.

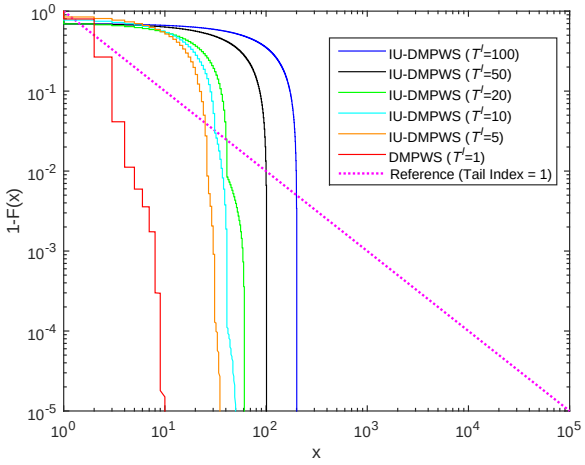


Figure 39. Packet delay of LT traffic under the IU-DMPWS policy with various update frequencies T^I.

of state measurements (i.e., from $T^I = 1$ to $T^I > 1$) by simulating the IU-DMPWS policy. Figure 38 shows the performance of IU-DMPWS policies with $T^I = 2$. IU-DMPWS can provide bounded queue lengths and delay for LT flows and achieve the stable network. It is consistent with Theorem 7 that such an infrequent update will not affect the throughput optimality and network stability. Moreover, we further compare the packet delay under IU-DMPWS policies with various update frequency in Figure 39. It is shown that as the update frequency increases, the delay of LT flows will also slightly increase but still have finite mean, which confirms our theoretical finding. These outstanding performances accompanied with infrequent updating facilitate practical implementation of IU-DMPWS.

5.2.5 Highlights

This section develops a throughput-optimal and delay-based maximum power-weight scheduling in switch/wireless hypervisors for single-hop flows with heavy-tailed traffic. Delay-based scheduling provides a simple way to reduce packet delay that plagues its queue length-based counterpart. We propose a DMPWS and its variant IU-DMPWS and prove their throughput optimality with respect to moment stability. The IU-DMPWS policy is favored among these scheduling algorithms for practical implementation because of its immunity to the impact of HT traffic along with limited signaling overhead.

5.3 Delay-based Throughput-optimal Scheduling with Heavy-tailed Traffic for 5G SoftAir: LIFO Policy

This section aims to develop novel throughput-optimal scheduling algorithms under hybrid HT and LT traffic flows, where classic optimal policies (e.g., maximum-weight/backpressure schemes), developed under LT assumption, are not throughput-optimal anymore. To counter this problem, a delay-based maximum-weight scheduling policy with the last-in first-out (LIFO) service discipline, namely LIFO-DMWS, is proposed with the proved throughput optimality under hybrid HT and LT traffic. The throughput optimality of LIFO-DMWS gives that a networked system can support the largest set of incoming traffic flows, while guaranteeing bounded queueing delay to each queue, no matter the queue has HT or LT traffic arrival. Specifically, by exploiting asymptotic queueing analysis, LIFO-DMWS is proved to achieve throughout optimality without requiring any knowledge of traffic statistic information (e.g., the tailness or burstiness of traffic flows). Simulation results validate the derived theories and confirm that LIFO-DMWS achieves bounded delay for all flows under challenging HT environments.

5.3.1 Motivation and Related Work

Because of the emerging multimedia, data center, and the Internet applications over mobile devices, network traffic in both radio access networks and wired core networks has tremendously grown in past few years while the network capacity is rather limited. To address such a challenge, throughput-optimal scheduling is highly demanding, which determines the optimal transmission time for traffic flows, supports the largest set of traffic rates, and maintains the desired network stability. As being an important class of throughput-optimal scheduling, the maximum-weight scheduling (MWS) policy and many of its variants [84] are of great interests. However, it is proved that the celebrated MWS policies are not throughput-optimal anymore in the presence of HT traffic flows because MWS can lead to unbounded average queueing delay even if the arrival traffic rates are within network capacity region [92].

To counter above challenges, in this section, we propose a delay-based maximum-weight scheduling policy with LIFO service discipline (LIFO-DMWS) and prove its throughput optimality in the presence of HT traffic. Specifically, rather than adopting queue backlog as link weight, we focus on delay-based scheduling, which exploits the *head-of-line (HoL) delay* metric in inter-queue scheduling decisions (i.e., determining the serving order for the packets from different queues). Moreover, instead of using the classic FIFO service discipline, we exploit LIFO service discipline for intra-queue scheduling (i.e., determining the serving order for the packets within each queue). Furthermore, by exploiting asymptotic queueing delay analysis along with moment theory, we prove that LIFO-DMWS is throughput-optimal with respect to strong stability in the presence of heavy tails. That is, we show that with LIFO-DMWS, no matter the incoming traffic flows are HT or LT, all queues will experience bounded average queueing delay as long as the incoming traffic rates are within the network capacity region. Such a throughput optimality feature is of great importance, since it prevents the QoS performance of LT traffic from being significantly degraded by the bursty HT traffic. Simulation results confirm the throughput optimality of LIFO-DMWS and show that LIFO-DMWS brings considerable delay reduction as compared to classic maximum-weight scheduling policies [7]. To the best of our knowledge, this work is the first throughput-optimal scheduling for bounded delay with emerging HT traffic.

5.3.2 LIFO Delay-based Maximum Weight Scheduling (LIFO-DMWS)

The queueing system model follows the details in Chapter 5.2.2. Moreover, by properly selecting α to allocate more service opportunities to LT queues, DMPWS in Chapter 5.2 can guarantee that all LT queues experience bounded average queueing delay, completely shielding those LT queues from the destructive impact of HT traffic. However, DMPWS cannot ensure the delay boundness of the HT queues and is still not a throughput-optimal scheduling policy with respect to strong stability. Moreover, DMPWS policy requires the statistical tailness information, which is difficult to estimate. To counter above challenges,

we propose LIFO delay-based maximum weight scheduling (LIFO-DMWS) policy [93]. Different from the classic MWS policies, the proposed LIFO-DMWS exploits LIFO service discipline for intra-queue scheduling and employs the HoL delay as the weight for inter-queue scheduling. At each time slot, the LIFO-DMWS policy serves the queue f with the maximum HoL delay, i.e.,

$$W_f^{LIFO}(t) = \max_{j \in F} W_j^{LIFO}(t), \quad \forall f \in F. \tag{67}$$

where ties are broken randomly.

Theorem 8 (Throughput Optimality of LIFO-DMWS [93]) *LIFO-DMWS is throughput optimal by ensuring that all queues have bounded average queueing delay* $E[D_f] < \infty, \forall f \in F$ *whenever (1) all arrival traffic flows have bounded mean* $\lambda_f = E[A_f(t)] \in \infty, \forall f \in F$. *That is, all arrival traffic flows have a tail index larger than 1, i.e.,* $\min_{f \in F} \kappa(A_f(t)) > 1$*; (2) the incoming traffic rates are within the network capacity region.*

Intuitively speaking, the promising feature of LIFO-DMWS comes from the optimal asymptotic delay performance of the HT queues under LIFO service discipline, which can guarantees that HT queues experience sufficiently reduced delay, which are of bounded mean. Then, by adopting HoL delay as the weight metric, we can ensure that the delay of LT queues is at least of the same order as the delay of HT queues. This implies the queueing delay of LT queues is also of the bounded mean. Accordingly, we can show that LIFO-DMWS is throughput-optimal. What is more important, LIFO-DMWS does not require any knowledge of the statistical information of traffic arrivals.

Despite the intuitive advantage of LIFO-DMWS, proving its throughput optimality with respect to strong stability is very challenging. We adopt asymptotic queueing analysis [94]. We first investigate the asymptotic queueing delay performance under the general *work-conserving* scheduling policies in Lemma 3, which gives the upper bound of the tail index for queueing delay under LIFO-DMWS. In Lemma 2, we derive the asymptotic delay performance under LIFO-DMWS, which yields the lower bound of the tail index for queueing

delay. Then, we prove the throughput optimality of LIFO-DMWS by showing that the upper and lower bounds coincide and are larger than 1 as long as the arrival traffic has tail index larger than 1. This indicates that all queues are of bounded average queueing delay under LIFO-DMWS. Before going to the main theorems, we first introduce Lemma 1.

Lemma 1 ([93]) *For any work-conserving scheduling with single-hop hybrid traffic, we have the following: under LIFO,* $\kappa(\max_{f\in F} D_f) \geq \min_{f\in F} \kappa(A_f(t))$.

Lemma 2 ([93]) *Consider any work-conserving scheduling with arrival traffic satisfying* $\min_{f\in F} \kappa(A_f(t)) > 1$. *Under LIFO, the steady-state delay* D_f *follows* $\kappa(A_f(t)) \geq \kappa(D_f) \geq \min_{f\in F} \kappa(A_f(t))$ *whenever incoming traffic rates are within the network capacity region.*

Now, we derive the upper bound of the tail index of queueing delay.

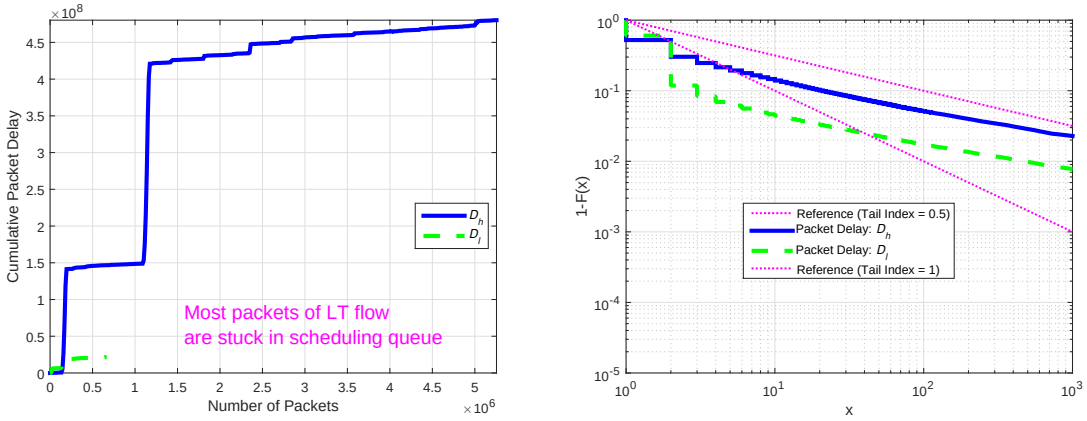
Lemma 3 ([93]) *Under LIFO-DMWS, the tail index* $\kappa(D_f)$ *of the steady-state queueing delay* D_f *is upper bounded by* $\kappa(D_f) \leq \min_{f\in F} \kappa(A_f(t))$.

Upon this stage, the throughput optimality of LIFO-DMWS is presented.

Proof of Theorem 8 [93]: Given that incoming traffic rates are within the network capacity region, this implies that the network is steady-state stable. Hence, we are ready to apply the asymptotic queueing analysis for LIFO-DMWS as follows. First, the upper bound of $\kappa(D_f)$ under LIFO-DMWS is given by Lemma 3. Since LIFO discipline is work-conserving, by Theorem 2, the lower bound of $\kappa(D_f)$ under LIFO-DMWS is obtained, i.e., $\kappa(D_f) \geq \min_{f\in F} \kappa(A_f(t))$. Therefore, it follows that the upper and lower bounds of $\kappa(D_f)$ coincide under LIFO-DMWS, i.e., $\kappa(D_f) = \min_{f\in F} \kappa(A_f(t))$. This, combing with the given condition $\min_{f\in F} \kappa(A_f(t)) > 1$ and Eq. (63), completes the proof.

5.3.3 Performance Evaluation

We show that bounded delay and throughput optimality can be achieved for hybrid HT and LT traffic by applying LIFO-DMWS. We choose Pareto and Poisson distributions to depict HT and LT distributions, respectively, as before. We consider a scenario where a HT flow



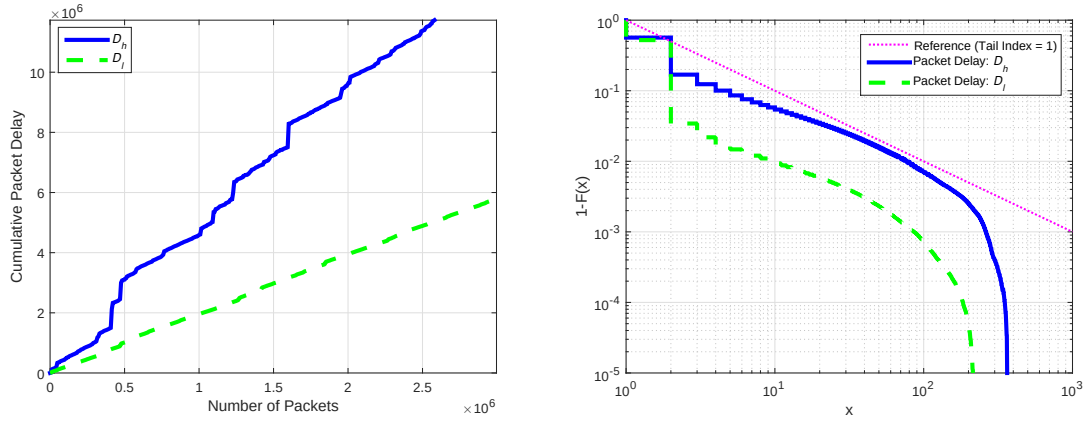
(a) Dynamics of cumulative packet delay.

(b) Tail distribution of packet delay.

Figure 40. Queueing delay of a HT flow, i.e., $A_h(t) \in \mathcal{PAR}(1.5, 1)$, and a LT flow, i.e., $A_l(t) \in Poiss(3)$, under (FIFO-)DMWS [7].

and a LT flow sharing a single channel, i.e., $A_h(t) \in PAR(1.5, 1)$ and $A_l(t) \in Poiss(3)$. All the following tail distribution results are plotted on log-log coordinates, by which a HT distribution with tail index κ manifests itself as a straight line with the slope equal to $-\kappa$. We first examine the performance of hybrid traffic under the classic MWS policy. To enable a fair comparison, we also adopt delay as the weight metric for MWS instead of queue length and denote it by (FIFO-)DMWS. Figure 40(a) shows that during 10^5 time slots, only around 10% of packets (as compared to HT traffic) from LT traffic leave the scheduler and contribute to the cumulative packet delay, given the same packer arrival rate for both LT and HT flows. The reason is that most of packets from LT traffic are stuck in the queue due to the competitions with the HT flows. Moreover, Figure 40(b) shows that under the DMWS policy, the queueing delay of LT flow follows heavy tailed distribution with a tail index smaller than one, as its tail distribution decays slower than the reference Pareto distribution with tail index one. This means that the LT traffic also has unbounded average delay as that of HT traffic. Hence, under the DMWS policy with hybrid traffic, the LT flow necessarily has infinite average queueing delay, and DMWS is not throughput optimal.

We next show that the bounded delay and thus throughput optimality can be achieved



(a) Dynamics of cumulative packet delay.

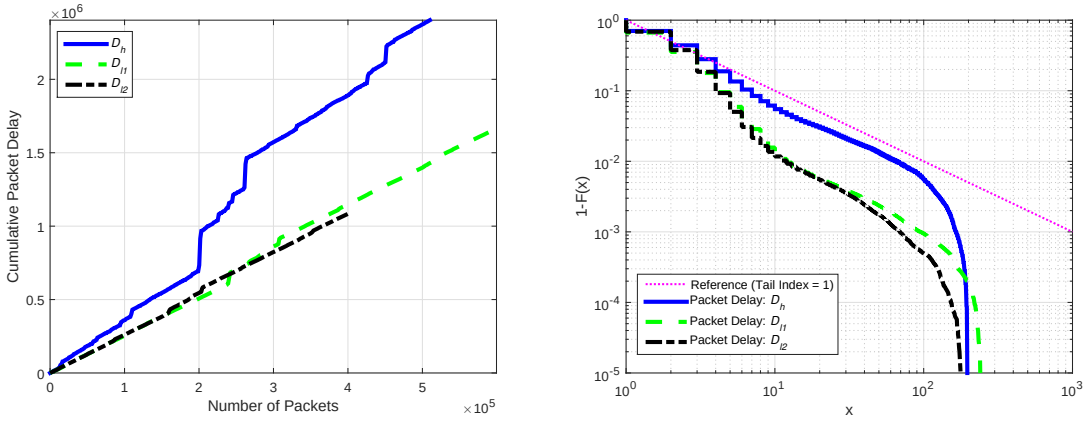
(b) Tail distribution of packet delay.

Figure 41. Queueing delay of a HT flow, i.e., $A_h(t) \in \mathcal{PAR}(1.5, 1)$, and a LT flow, i.e., $A_l(t) \in Poiss(3)$, under LIFO-DMWS.

for hybrid traffic by applying the LIFO-DMWS policy. Figure 41(a) shows that LT traffic can receive sufficient service opportunities to sent a comparable number of packets as HT traffic flows. Moreover, as shown in Figure 41(b) that under the LIFO-DMWS policy, the queueing delay of LT flow has a tail index greater than one, as their tail distributions decay faster than the reference Pareto distribution with tail index one. This means the LT traffic has bounded queueing delay. Furthermore, while the queueing delay of HT flows are also of bounded mean. Finally, a more complicated scenario under LIFO-DMWS is studied with three hybrid flows in Figure 42, i.e., $A_h(t) \in PAR(1.5, 1)$, $A_{l1}(t) \in Poiss(3)$, and $A_{l2}(t) \in Poiss(2)$, and the same conclusion is reached accordingly. In particular, Figure 42(a) shows that most of packets from LT flows can exit the system. Figure 42(b) further indicates that for the tail distributions of queueing delay of all LT and HT flows have a slope or decaying rate greater than one, indicating the average bounded delay for all traffic flows. Above results are consistent with Theorem 8, which implies that under hybrid HT and LT traffic, LIFO-DMWS is throughput-optimal by achieving strong stability.

5.3.4 Highlights

LIFO-DMWS is proposed to achieve throughput-optimality with heavy-tailed traffic. In particular, LIFO-DMWS guarantees bounded average delay for hybrid HT and LT flows



(a) Dynamics of cumulative packet delay.

(b) Tail distribution of packet delay.

Figure 42. Queueing delay of a HT flow, i.e., $A_h(t) \in \mathcal{PAR}(1.5, 1)$, and two LT flows, i.e., $A_{l1}(t) \in Poiss(3)$ and $A_{l2}(t) \in Poiss(2)$, under LIFO-DMWS.

under any admissible traffic arrivals without any knowledge of statistical information of the arrivals. Performance evaluations validate our theoretical findings. The future research will be the extension of LIFO-DMWS to multi-hop traffic flows with possibly feedback loops.

5.4 QoS-aware Adaptive Routing in Distributed Hierarchical SoftAir Systems

SDNs have been recognized as a next-generation networking paradigm that decouples data forwarding and control command. These decoupled SDNs bring centralized control over the entire system, and enable dedicated QoS provisioning and fast route (re-)configuration services. To this end, diverse QoS requirements from user applications in packet delay, loss, and throughput should be supported by an efficient data transportation. In this section, a QoS-aware adaptive routing (QAR) algorithm is proposed as network service in the designated multi-layer hierarchical SDNs. The distributed hierarchical control plane architecture is first exploited to minimize signaling delay in large SDNs, through three-levels design of controllers (i.e., the *super, domain or master, and slave controllers*). Furthermore, upon the hierarchical control plane, the QAR algorithm is proposed with the aid of reinforcement learning and QoS-aware rewards, achieving time-efficient, adaptive, and QoS-provisioning

packet forwarding. Numerical results show that QAR outperforms the existing Q-learning solution and provides fast convergence with guaranteed QoS, thus facilitating the practical implementation in large-scale SDNs or software service-defined networks.

5.4.1 Motivation and Related Work

SDNs have the promise to dramatically improve network resource utilization, simplify network management, reduce operating cost, and promote innovation and evolution [2, 3, 19]. However, despite of their "advertised" promising features, a reliable end-to-end data transportation upon SDNs is hard to design due to the requirements of diverse quality-of-service (QoS) provisioning and fast route (re-)configuration. In particular, aiming at upholding a great variety of applications, SDNs should fulfill various QoS requirements such as in packet delay, packet loss, and throughput. Moreover, as users and multi-tenancy applications greatly increase as well as network topology and traffic statistic change over time, SDNs should also provide a fast and adaptive data transportation in order to react such events in a real-time manner. Thus, it becomes a great challenge and also an urgent need to support a time-efficient and QoS-aware routing service in large-scale SDNs or software service-defined networks.

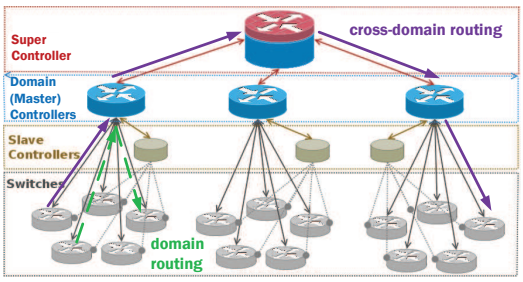
While the specification of OpenFlow [36] requires the logically centralized control plane in SDNs, single controller scheme faces several crucial issues such as network scalability, single-point failure, and frequent reporting control tasks, per-flow data supervision. Recent work [95–97] focuses on distributed multi-controller platforms. In Onix [95], a horizontally flat structure of all controllers is proposed and a general management API is implemented to connect multi-controllers. In [98], a scalable clustering approach and a self-learning adaptive mechanism are proposed to design and implement SDN controller clusters. In Kandoo [96], a completely hierarchical model is proposed, including a logically centralized root controller and local controllers, and local controllers can directly offload the applications that do not need network-wide information to the underlying switches. In Xbar [97], a recursive hierarchy design is proposed among multiple controllers that a

lower-level controller is recognized as a switch by its upper controller. Facing these distributed SDN systems, an adaptive routing that exploits the system advantages and fulfills QoS requirements in a timely manner is still unexplored.

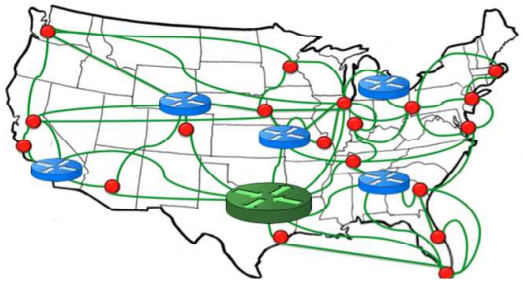
In this section, we propose a QoS-aware adaptive routing (QAR) algorithm in our designated multi-layer hierarchical SDNs. First, inspired by the work of Kandoo [96] and Xbar [97], a distributed hierarchical control plane architecture is introduced that combines the advantages of both work and is complied with OpenFlow 1.2+. Specifically, the three hierarchical levels of distributed controllers, including the *super, domain (or master), and slave controllers*, and the switch subnets (i.e., clustering) are proposed. Exploiting such a novel architecture, the control loads are shared and the signaling delay can be largely reduced. Furthermore, with the aid of reinforcement learning (RL) [8] and our preliminary study [99], QAR algorithm is proposed through the examination of an action rule, quality function, long-term revenue, and system model with a reward function. Specifically, the softmax action selection rule, state-action-reward-state-action (SARSA) [100] method for quality update, and Markov decision process (MDP) with QoS-aware reward function are introduced to realize an efficient, adaptive, and QoS-provisioning routing algorithm.

Inherited from RL, QAR enables four crucial features:

- It has fast adaptation to the current network and traffic states for the time-varying multi-tenancy environment as well as network topology.
- It well distributes the traffic loads from QoS-aware rewards, avoiding congestions as in shortest-path algorithms.
- It has great scalability due to scalable learning, easily including new devices in next learning iteration.
- It supports customized requirements from its tunable parameters in and generic design of rewards.



(a) Distributed hierarchical architecture.



(b) A deployment example in a real Sprint GIP network [5] over North America.

Figure 43. Multi-layer hierarchical control plane.

A rigid convergence analysis shows that QAR, based on the RL framework, has a sublinear convergence rate, verifying fast adaptation of QAR to network dynamics. Moreover, performance evaluation confirms that QAR outperforms the conventional Q-learning [8] approach with great time-convergence and QoS provisioning in real backbone networks. To the best of our knowledge, this work is the first to provide a QoS-aware adaptive routing algorithm with preferred fast-convergence, through RL, in multi-layer hierarchical SDNs.

5.4.2 Multi-layer Hierarchical Control Plane Architecture

Based on the architecture designs of Kandoo [96] and Xbar [97], we propose a multi-layer hierarchical control architecture that combines the advantages of these two existing solutions and is complied with OpenFlow protocol [36]. The details of the proposed hierarchical control solution are explained as follows. The proposed architecture consists of a recursive hierarchical control plane with three levels of controllers as shown in Figure 43(a). While switches take charge of data forwarding and information collection of network status, the slave controllers provide read-only access to switches and receive port-status messages from them. These slave controllers not only serve as the message dispatchers as in [98] that ease the bottleneck of excessive control messages in the I/O frontend, but also can provide some simple control functions, such as traffic admission control, flow or congestion control, to share control workloads with domain controllers. Moreover, the domain (or master) controllers, having full accesses to switches, receive asynchronous messages

(e.g., Packet-IN [36]) for flow-setup requests and are capable to modify switches' states by sending control messages. Finally, the only one super controller, connecting to domain controllers, also has full accessibility to switches and regulates the entire network functionalities. The interaction between super controller and domain controllers fulfills global flow setup and responds to every control action, including actions for switches' or controllers' failures, migrations, load-balancing, etc. Furthermore, the slave controllers only aim to offload control messages for various applications and do not require network-wide information, largely saving the signaling overheads. Towards this, the logically centralized control plane with global visibility is established by a physically distributed system. All applications running upon SDN are unaware of the underlying distributed architecture.

A deployment example in a real backbone system of Sprint GIP network [5] is illustrates in Figure 43(b), where a red spot denote a slave controller with a group of switches underneath, a blue device denotes a domain controller serving a switch subnet and possibly more than one slave controller, and the green device denotes the super controller to supervise the entire system. Specifically, in [42], we provide an effective mechanism to choose how many and where to place those controllers and how control messages should be routed over the same in-band network such that there is minimum acceptable interference to data traffic. One advantage of the design is that this architecture decouples the failure recovery from path computation. In particular, the super controller designates a domain controller whenever a failure occurs, and the domain controller computes the corresponding recovery path to resolve the failure. What is more important, to ensure reliability and robustness upon controllers' failure, switches at the edge of a subnet can establish communication with more than one domain controller. In that way, it allows easy control handover between switch subnets with respect to their domain controllers. Furthermore, with full accessibility to switches, the super controller and domain controllers have the ability to identify the cause of switch asynchronization and enable/disable the warning notification.

In spite of these considerable advantages, to realize an efficient adaptive routing upon

the proposed architecture, the hierarchical structure is exploited to greatly minimize the signaling delay between controllers and switches. The idea comes from (i) the signaling-load distribution and (ii) parallel path computation. Specifically, each domain controller is in charge of the signaling within its own switch subnet in such a way when the first packet of a flow arrives into a specific subnet, the optimal route is solely computed by the corresponding domain controller. Only when the destination switch of arriving flow is outside the source switch's subnet, the packet is then forwarded to the super controller and multiple subnets (and domain controllers) will be involved in the path computation. In particular, the super controller exclusively calculates the subnet-path, i.e., the set of subnets that the packet flow will go through to reach its destination, with the aid of its global visibility. Once the subnet-path is calculated, the super controller forwards the control messages to the involved domain controllers in order to active their own path computations within the respective subnet. As the optimal paths in different subnets can be computed in parallel, the flow does not need to wait whenever it enters the next subnet along the calculated subnet-path, thus minimizing the entire signaling delay.

5.4.3 QoS-aware Adaptive Routing (QAR)

Based on the designated distributed hierarchical control plane, we propose the QAR algorithm with the aid of RL technique. We first introduce the learning framework, provide our design of QoS-aware reward functions, and finally propose QAR.

Reinforcement Learning Framework: reinforcement learning (RL) [8], as a field of machine learning, captures the problem that an agent/decision maker tries to learn the behavior of dynamic system through the interactions with the system. Specifically, at each iteration, the agent receives the current state and the reward from the dynamic system, and then performs the respective action according to its pat experience in order to increase the long-term revenue via state transitions. The state and the reward are the two values that the agent will receive from the system, whereas the action is the only input that the system will receive from the agent. Different from the supervised learning techniques with an external

knowledge supervisor, in RL, the agent must discover the best action that maximizes the reward itself. This reward value indicates the success of agent's action decisions, and the agent learns which actions to be selected to provide the highest accumulated reward over time, i.e., the long-term revenue. While agent's actions affect not only the immediate reward but also the subsequent one, the key feature of RL is to perform incentive solution searching regarding system rewards. To realize this searching in an efficient way, the design challenge comes from the balance between action exploration and action exploitation, where such a trade-off is well studied in [101]. In particular, the agent has to exploit the past actions with great rewards, and to explore the system for better unknown actions at the same time. It means both the exploitation and exploration needs to be pursued conjointly for the optimal system performance. As there are many sophisticated algorithms that detail this joint consideration into their trial-and-error designs, the RL only characterizes the interaction procedures instead of providing another learning methods. In other words, any learning algorithm can be seen and transformed into RL. In the following, based on the learning technique, we provide several design ingredients that are used by our adaptive, time-efficient, and QoS-aware routing.

Markov Decision Problem: States, Actions, and Rewards. In addition to an agent/decision maker and the dynamic system, there are four main ingredients for the RL design: (i) the action rule, (ii) the quality function, (iii) the long-term revenue, and (iv) the system model with reward functions. Specifically, the action rule is the decision that will be taken by the agent. It is the mapping from the perceived system states to the corresponding actions, and guides RL behaviors. Moreover, the quality function characterizes the quality of each state-action pair that indicates the differences between current state and steady state. Furthermore, the long-term revenue indicates the total rewards an agent can expect to accumulate over time with respect to each system state. Whereas the reward is given after each agent's current action, this revenue shows the long-term desirability of states regarding the future states that will be followed and their respective rewards. Last, the system model

mimics the behavior of the real environment system, and gives a good reward prediction of the next state and quality from the current ones. To this end, the MDP provides a systematic modeling for the RL design. A MDP is denoted by the quadruple (S, A, P, R), where S denotes the finite state set, A the finite action set, P the set of state transition probabilities, and R the reward set. We have exploited this MDP framework into our routing system with the context of learning a routing strategy. We consider each switch along the routing path as a state and each link emerging from a switch as an action to choose. The routing mechanism then corresponds to the control policy that the agent will learn. Regarding QoS-awareness, we further propose QoS-aware rewards that will be evaluated for all links and facilitate RL with the optimal routing path over the network. The proposed routing mechanism in a switch subnet is illustrated as follows:

- The source switch sends a new data flow to the destination switch. For each link transmission, the controller (i) calculates the QoS-aware reward, (ii) executes the action selection process for the next-hop, and (iii) updates the Q-value of the current link. According to the selected routing path, the controller then installs the forwarding rules in the flow tables of intermediate switches. These rules will be followed by the switches for their next-hop transmissions, up to the destination switch.

Note that in [102], while an end-to-end QoE model is also investigated with the designed RL-based routing, two sets of flow transmissions (i.e., data and ACK flows) are needed to complete the learning process. This easily causes great signaling overheads (i.e., ACK messages for QoE values) and possibly divergent routing strategies (e.g., the data path might be unavailable for ACK transmissions due to node or link failure.) On the other hand, with the aid of centralized control and global information accessibility in SDNs, we are able to not only achieve one-way (i.e., data flow) RL-based routing, but also retain optimal routing decisions by real-time adjusting route selection in regard to dynamic network conditions. In the following, we examine the details of selection rules, the learning process with quality functions, and reward designs for our routing mechanism.

Action Selection Rule: the selection rule specifies an agent's action selection and maps the state to the action. It balances the trade-off between action exploitation and exploration to maximize the quality value, as the agent aims to *explore* the state space in the beginning. Three policies are widely-used, i.e., the greedy, the ϵ-greedy, and the softmax [103]. For the greedy rule, the agent takes the action with the highest quality at every step. It simply exploits the current agent's knowledge base of state and quality function, and does not explore unknown states for possibly higher quality. This makes the rule undesirable when the quality function is non-stationary, which changes over time. Moreover, ϵ-greedy balances the current knowledge exploitation with action exploration that follows the greedy rule with probability $1-\epsilon$ and takes a random action with probability ϵ. It serves as an effective rule when there are a great variety of possible actions. However, the drawback of ϵ is that when it explores it chooses equally among all actions. This implies that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action, which is unsatisfactory when the worst actions are very bad.

Towards this, we consider softmax and adopt it in our designated routing. Specifically, regarding softmax, the probability $\pi_t^{(i,j)}(s,a)$ of choosing an action $a_t^{(i,j)}$ (i.e., switch i selects switch j as the next-hop) with the current state $s_t^{(i,j)}$ (i.e., switch i along the routing path) follows

$$\pi_t^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)}) = \frac{\exp\left(\frac{Q_t^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)})}{\tau_t}\right)}{\sum_{b=1}^{A(i)} \exp\left(\frac{Q_t^{(i,j)}(s_t^{(i,j)}, b_t^{(i,j)})}{\tau_t}\right)}, \forall 1 \leq j \leq A(i) \tag{68}$$

where $A(i)$ denotes the number of switch i's neighbors, $Q_t^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)})$ denotes the corresponding quality function, and τ_t is a parameter called temperature. This time-varying temperature controls the trade-off (balance) between exploitation of current states and exploration of future states. High temperature causes all actions to be equally probable (i.e., exploration), while low temperature favors the action with the maximum quality (i.e., exploitation of current knowledge base) that skews the rule toward a greedy one. In this way, the temperature parameter is annealed over the training phase that has greater exploration

in the beginning and greater exploitation near the end. It means τ_t remains a high value in highly dynamic environments while decreases towards a low value in static environments where the convergence is assured. To achieve the learning convergence in finite time, the temperature is thus set as a linear function over time as

$$\tau_t = -\frac{(\tau_0 - \tau_T)t}{T} + \tau_0, \quad t \leq T, \tag{69}$$

where T denotes the time to reach the convergence, and τ_0 and τ_T are the initial temperature and last temperatures at time T, respectively. It implies that $\tau_t \neq 0$ for all time and $\tau_t = \tau_T \approx 0$ for $t \leq T$ until any system-parameter change.

State-Action Quality Function: in addition to estimate the outcome quality solely by the possible next system state, the agent can set up its quality function based on both the state and action. Specifically, the quality function $\mathbf{Q}(\mathbf{s}, \mathbf{a})$ is introduced that shows the quality for taking action $\mathbf{a}$ at current state $\mathbf{s}$. When the agent needs to choose an action for current state, it simply calculates the quality function for each possible action and chooses the next action according to these quality values. With the context of learning the routing strategy, once the QoS-aware reward of a link is calculated, the controller will update the Q-value of the link according to the policy. In the following, two methods for setting quality function are provided as the conventional Q-learning [8] and SARSA [100], and SARSA is adopted in our designated routing.

First, the well-known Q-learning, being as off-policy RL (i.e., a learner learns the value of the optimal policy independently of the agent's actions), updates Q-function as follows

$$Q_{t+1}^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)}) := (1-\alpha) Q_t^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)}) + \alpha \left[R_{t+1}^{(i,j)} + \gamma \max_a Q_t^{(i,j)}(s_{t+1}^{(i,j)}, a) \right] \tag{70}$$

where $\gamma \in [0, 1)$ is the discount factor that determines the importance of future rewards, $\alpha \in [0, 1)$ is the learning rate that determines the override extent of the newly acquired information to the old one, and $R_{t+1}^{(i,j)}$ is the immediate reward from selecting link $i \rightarrow j$ at time $t+1$. Parameters α and γ should be designed wisely according to specific applications for preventing learning diverges. In Eq. (70), the agent updates the quality based on the

maximum possible quality value among its actions. Specifically, the agent chooses and takes action $a_t^{(i,j)}$ for current state $s_t^{(i,j)}$ via action selection rule, observes $R_{t+1}^{(i,j)}$ and state $s_{t+1}^{(i,j)}$, and Q-function can be updated accordingly. The shortage of Q-learning comes from the updating assumption that the optimal action with the maximum Q-value is always selected, no matter what the agent actually does. This ignorance of agent's actual actions might danger the system operation due to possibly great negative rewards. On the other hand, regarding on-policy RL (i.e., a learner learns the value of the policy being carried out by the agent, including the exploration steps) of SARSA, the quality function is updated by

$$Q_{t+1}^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)}) := (1-\alpha)Q_t^{(i,j)}(s_t^{(i,j)}, a_t^{(i,j)}) + \alpha\left[R_{t+1}^{(i,j)} + \gamma Q_t^{(i,j)}(s_{t+1}^{(i,j)}, a_{t+1}^{(i,j)})\right]. \quad (71)$$

In this time, the agent updates Q-function strictly on the knowledge base from the experience and iteratively improves the Q-value. Specifically in Eq. (71), different from Q-learning, the agent uses the action and the state at time $t+1$ to update quality value. Thus, the only difference between these two methods is the way they set up for the future reward. That is, whereas Q-learning utilizes the highest quality function at state $s_{t+1}^{(i,j)}$ regarding all possible actions, SARSA adopts the quality function at state $s_{t+1}^{(i,j)}$ with action $a_{t+1}^{(i,j)}$. General speaking, Q-learning simply assumes an optimal action rule will be followed in the future; SARSA utilizes the rule that the agent indeed follows in the future. It implies that the agent with SARSA explicitly adopts the future reward that is really obtained, rather than assuming the optimal action with highest reward will be taken.

QoS-aware Reward Design: we propose QoS-aware reward functions that suits our design of QoS-aware routing. Specifically, based on RL, the agent finds the routing path with the maximum QoS-aware reward with regard of traffic types and users' applications. TABLE 5 summarizes diverse QoS requirements of widely-applied traffic and applications.

Table 5. QoS Requirements With Respect to Traffic Types and Applications.

Traffic Type	Application	QoS-awareness
Elastic	Telnet connection; FTP session	Delay, losses
	Simple web page (HTTP)	Delay
	Heavy web page (HTTP)	Throughput
	STMP/POP3/IMAP	Losses
	FTP data connection	Throughput
	Data with Telnet	Losses
Inelastic	Real-time multimedia	Delay, throughput, jitter
	Control message	Delay

For example, real-time traffic, inelastic to adapt packet transmission rates, has great QoS-awareness among others. A QoS-aware reward function is proposed as follows:

$$\begin{aligned} R_{t+1}^{(i,j)} &:= R(i \to j|_{\mathbf{s}_t,\mathbf{a}_t}) \\ &= -g(a_t^{(i,j)}) - \beta_1(\theta_1 delay_{ij} + \theta_2 queue_j) + \beta_2 loss_j + \beta_3(\phi_1 B1_{ij} + \phi_2 B2_{ij}), \quad (72) \end{aligned}$$

which shows that the system at state s_t, receiving action a_t, forwards packets from switch i to switch j. In Eq. (72), $g(\cdot)$ denotes the cost to take action a_t that reveals the action impact to switch operations, and $\beta_1, \beta_2, \beta_3, \theta_1, \theta_2, \phi_1, \phi_2 \in [0, 1)$ are tuneable weights, determined by the QoS requirements of flow. Aiming at QoS provisioning, the cost g is set to a constant value over actions, and the QoS-aware functions in Figure 44 are defined as follows:

$$delay_{ij} = \frac{2}{\pi} \arctan(d_{ij}^l - \frac{\sum_{k=1}^{A(i)} d_{ik}^l}{A(i)}); \quad (73a)$$

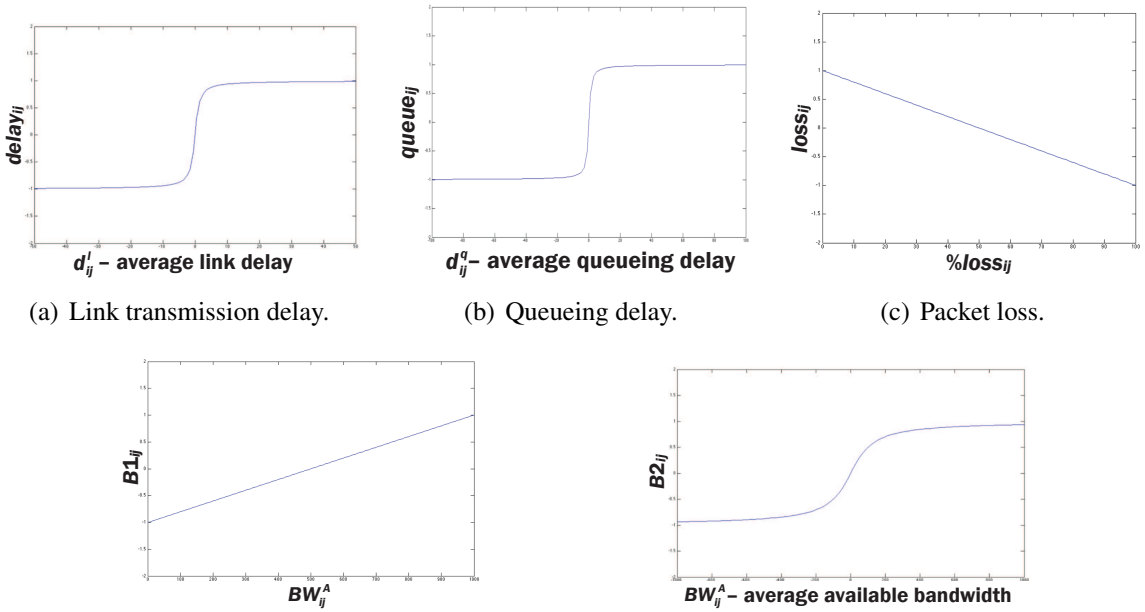
$$queue_{ij} = \frac{2}{\pi} \arctan(d_{ij}^q - \frac{\sum_{k=1}^{N} d_{ik}^q}{N}); \quad (73b)$$

$$loss_{ij} = 1 - 2\%\text{loss}_{ij}; \quad (73c)$$

$$B1_{ij} = \frac{2BW_{ij}^A}{BW_{ij}^T} - 1; \quad (73d)$$

$$B2_{ij} = \frac{2}{\pi} \arctan(0.01(BW_{ij}^A - \frac{\sum_{k=1}^{N} BW_{ik}^A}{N})), \quad (73e)$$

where d_{ij}^l and d_j^q are link transmission delay and packet queueing delay from switch i to



(a) Link transmission delay.

(b) Queueing delay.

(c) Packet loss.

(d) Available bandwidth in the next switch.

(e) Difference of available bandwidth in the next switch and average bandwidth.

Figure 44. QoS-aware reward functions.

switch j, respectively, N is the number of switches in the considered switch subnet, and $\%loss_{ij}$, BW^A_{ij}, and BW^T_{ij} characterizes the packet loss, available bandwidth, and total bandwidth of link $i \rightarrow j$, respectively. Eq. (73a) considers the link delay of link $i \rightarrow j$ comparing to other possible next hops, Eq. (73b) considers the queueing delay with respect to the average delay over the subnet, and Eq. (73c) characterizes the loss rate. Note that the different comparisons in Eq. (73a) and Eq. (73b), i.e., the neighbors $A(i)$ and the switches N in a subnet, respectively, indeed provides the thorough consideration of packet latency over the hierarchical structure and switch subnets. Eq. (73d) and Eq. (73e) indicate the available bandwidth of link $i \rightarrow j$ and that with respect to the average link bandwidth over the subnet, respectively. From Figure 44, all these QoS functions have the values within $[-1, 1]$, where the value closes to one means the link selection is preferred by the respective parameter and otherwise the value closes to negative one as the penalty.

QoS-aware Adaptive Routing (QAR): inspired by our previous study of adaptive computation framework for routing paths [104], we propose a QoS-aware adaptive routing

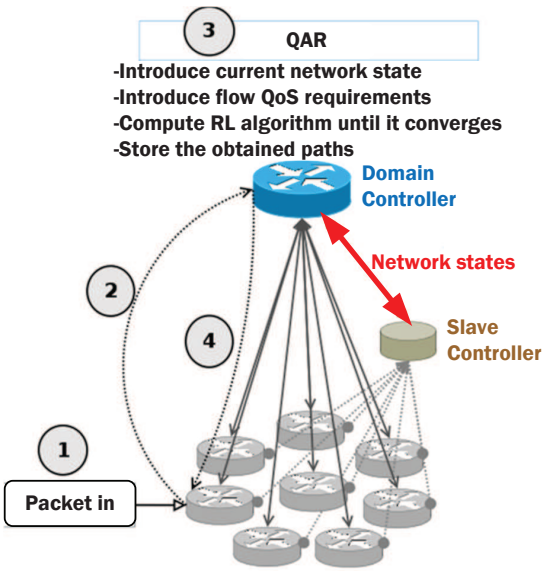


Figure 45. Flow diagram of QAR.

(QAR) in **Algorithm 8** with a flow diagram in Figure 45 through reinforcement learning and designated QoS-aware rewards in multi-layer hierarchical SDNs. Specifically, as mentioned in Section 5.4.2, this distributed SDN system consists of various switch subnets, and each subnet has a domain controller, one or many slave controllers, and many underlying switches. Moreover, domain controller takes charge of path calculation for each incoming flow, while slave controllers gather the network state and send the information updates to the domain controller shown in Figure 45. Thus, QAR determines the forwarding path inside each subnet for the respective domain controller and the global forwarding direction among subnets for the super controller. The procedures are explained as follows.

When a new flow arrives to a switch, the switch forwards the first packet of the flow to the domain controller and requests the forwarding path. The domain controller then updates the current network state regarding the latest information from slave controller(s), exploits the proposed RL in **Algorithm 9** to select a feasible path with respect to QoS requirements of the flow, and modifies the forwarding tables of switches along the selected path. Furthermore, if the domain controller realizes that the destination switch does not belong to its subnet, the first packet will also be sent to the super controller. The super

controller then performs **Algorithm 9** to find the forwarding direction among subnets (i.e., subnet-path), and send the corresponding notifications to the domain controllers of involved subnets. In this way, the RL executed by the super and domain controllers conjointly gets the global optimality. In particular, the path searching of these controllers can be executed simultaneously, thus saving much computation time to facilitate time-efficient QAR.

Note that the computation loads, distributed among three levels of controllers, are largely reduced through this hierarchical load-sharing. Moreover, being complied with OpenFlow, if there exists the matching entry of the incoming flow, the switch will not send the packet to domain controller but simply forwards the packet following the existing matching. Thus, QAR successfully provides a QoS-aware, time-efficient, and adaptive routing algorithm, particularly effective for large-scale SDNs. Finally, such a better routing function with the proper consideration of QoS requirements and changes of network status offers one of the best candidate for an innovative service-enabled, virtualized, and generalized network function that could be easily integrated with other network planning and management systems.

Algorithm 8: QoS-aware Adaptive Routing (QAR).

1 New flow f arrives to a switch in the subnet.
2 Switch forwards the first packet to domain controller E_f.
3 **if** $Dest(f)$ *is not in the same subnet* **then**
4 | Super controller executes **Algorithm 9**;
5 | Domain controllers along the subnet-path executes **Algorithm 9**;
6 **else**
7 | Domain controller E_f executes **Algorithm 9**;
8 **end**
9 The rest packets of flow are forwarded following the established flow tables in switches.

5.4.4 Performance Evaluation

In this section, we compare our proposed QAR with conventional learning algorithm in a practical Sprint GIP network [5] and multi-layer hierarchical controller architecture as in Figure 43(b). There are 25 deployed nodes as switches and 53 links, and actual link delay

Algorithm 9: Reinforcement Learning (RL) for QAR.

1 **Initialize** $\mathbf{Q}_0(\mathbf{s}_0, \mathbf{a}_0) = 0$ and $\mathbf{R}_0$ from Eq. (72).
At time t:
2 **Choose** action $\mathbf{a}_t$ according to softmax in Eq. (68).
3 **Observe** $\mathbf{s}_{t+1}$ and $\mathbf{R}_{t+1}$.
4 **Update** $\mathbf{Q}_{t+1}$ function according to Eq. (71).
5 Continue from step 2 at time $t := t + 1$.

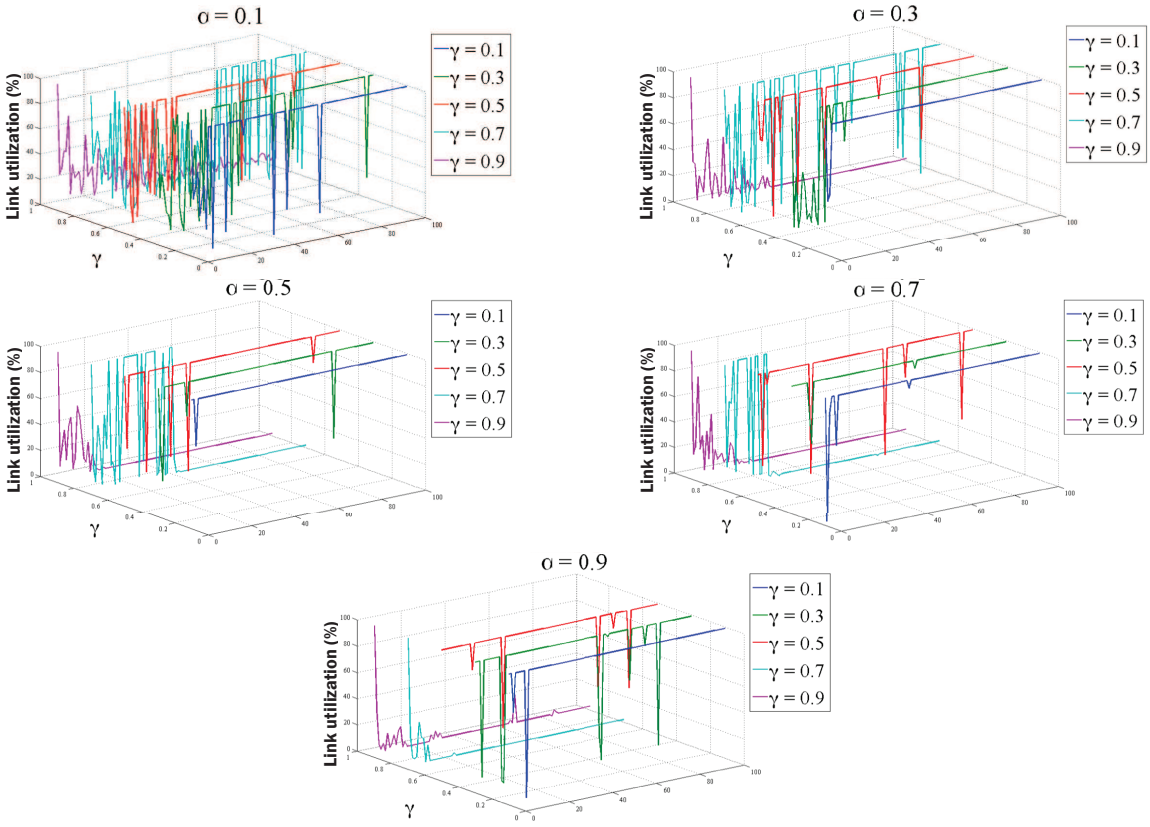
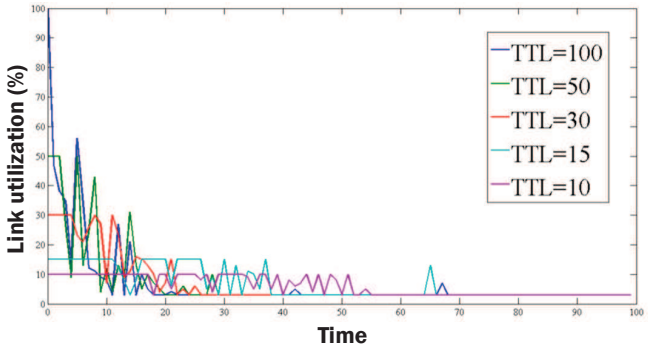


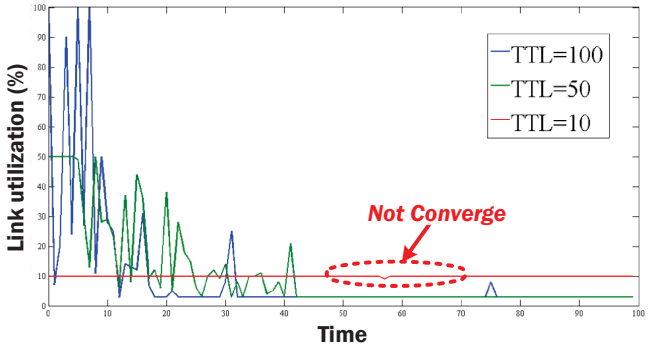
Figure 46. Link utilization with respect to different step-size parameters (α,γ) of QAR.

profiles are provided in [5].

Impacts of Step-Sizes in QAR Algorithm: based on RL, the performance of QAR is governed by the selection of step-size parameters (α,γ) as shown in Eq. (71). In particular, α adjusts the error that is included in the Q updating; $\gamma \in [0, 1)$ has zero value if the routing only considers the current reward and acts as greedy algorithm, and has the value close to one if the routing considers the long-term revenue. Figure 46 provides the percentage of link utilization of a suitable path via QAR with respect to (α,γ). It shows QAR converges



(a) Without QoS provisioning.

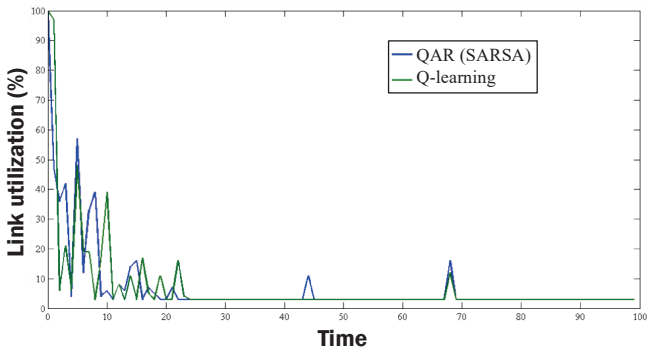


(b) With QoS provisioning.

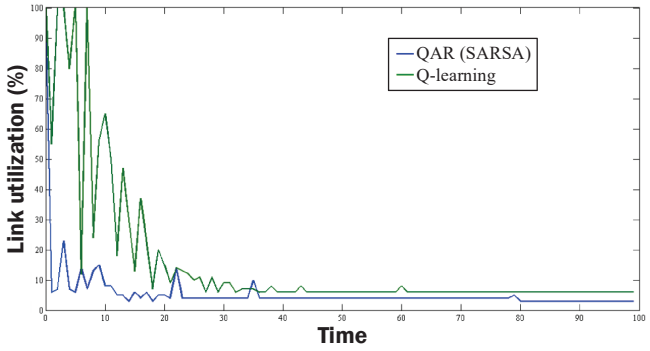
Figure 47. Time evolution of QAR.

only when $\gamma \in [0.7, 0.99]$, which implies the recent action indeed affects system performance more than the future ones. Moreover, with these large γ values, QAR performs better with few fluctuations when $\alpha \in [0.5, 1)$. Thus, these evaluations provide the preferred ranges of the step-size parameters with regards of routing performance, facilitating the practical implementation of QAR.

Performance Comparison of QAR Algorithm and Conventional Q-Learning [8]: we examine the time evolution of QAR without and with QoS provisioning, and compare QAR with the conventional Q-learning [8]. First, Figure 47 shows the QAR results with respect to different time-to-live (TTL) values that denote the maximum allowable number of searching iterations in each episode (TTL is set as 100). In particular, without QoS provisioning, the weights $\beta_1, \beta_2, \beta_3$ in Eq. (72) are set to zero, and Figure 47(a) shows that the greater the TTL is the faster the QAR converges. However, there exists a trade-off between



(a) Without QoS provisioning.



(b) With QoS provisioning.

Figure 48. Comparison between QAR and conventional Q-learning [8].

algorithm convergence and end-to-end delay. Specifically, while an increasing TTL brings better convergence, it also increases the computation time of each searching episode and thus increases end-to-end delay. Figure 47(b) further shows the results with QoS provisioning, where $\beta_1 = \beta_2 = \beta_3 = \theta_1 = \phi_1 = 1$ and $\theta_2 = \phi_2 = 0.5$. It indicates that the longer convergent time is required when considering QoS requirements. In particular, while QAR does not converge when TTL= 10, it requires double episodes to converge when TTL= 50. For TTL= 100, QAR takes almost the same episode to converge with or without QoS provisioning. Note that the fluctuations after the graph converges means that even finding a suitable forwarding path, QAR still tries to find the better solution with greater reward. Next, Figure 48 shows the comparison between QAR and Q-learning. It indicates that both approaches have similar convergent performance without QoS provisioning, but QAR outperforms with QoS provisioning. Thus, we successfully provide an effective and efficient

routing algorithm upon multi-layer hierarchical architecture for large-scale SDN.

5.4.5 Highlights

In this section, QAR is proposed as network service via RL in multi-layer hierarchical SDNs. This distributed hierarchical control plane is first introduced to minimize the signaling delay, serving as a realistic SDN architecture. QAR algorithm is then proposed to enable adaptive, time-efficient, and QoS-aware packet forwarding upon the proposed architecture. Performance evaluation confirms that QAR outperforms the conventional Q-learning approach with fast sublinear convergence when considering QoS provisioning. We have provided a novel network service that facilitates on-line and QoS-aware routing in practical implementation of large-scale software service-defined networks.

CHAPTER 6

CONCLUSIONS

In this thesis, we propose SoftAir as a new paradigm towards 5G wireless networks. SoftAir provides high flexible architecture, which can accelerate the innovations for both hardware forwarding infrastructure and software networking algorithms through control and data plane separation, enable the efficient and adaptive sharing of network resources through network virtualization, achieve maximum spectrum efficiency through cloud-based collaborative baseband processing, encourage the convergence of heterogeneous networks through open and technology-independent interfaces, and enhance energy efficiency through the dynamic scaling of computing capacity of the SD-BSs. First, we gave an detailed overview of state-of-the-art W-SDN architectures. Second, we introduced SoftAir architecture and four key design elements. Third, to realize the promising properties of SoftAir, we proposed the essential management tools, including control traffic balancing, optimal network planning, resource-efficient network virtualization, and QoS-aware traffic classifier. Fourth, we presented the novel software-defined traffic engineering solutions, including dynamic BS formation, throughput-optimal scheduling, and QoS-aware adaptive routing.

The contributions in each chapter are summarized as follows:

- In Chapter 2, we gave a detailed overview of priori W-SDN studies. We mentioned the major problems of scalability challenges and vendor-specific device configuration, facing by the current cellular architectures (e.g., LTE and LTE-A). We summarized the existing wireless work that considers SDN and NFV solutions and contributes to W-SDN trends. Moreover, we investigated several integrated W-SDN and NFV solutions and pointed out the missing parts in all related studies.

- In Chapter 3, we introduced SoftAir architecture and provided a completed qualitative comparison between SoftAir and existing architectures. In particular, four key design elements of SoftAir architecture were presented as scalable software-defined

planning, fine-grained fronthaul network decomposition, seamless OpenFlow incorporation, and network virtualization capacity. Also, the qualitative comparison of existing W-SDN solutions and SoftAir are provided with regard to architecture, scalability, network virtualization, traffic engineering solutions, and research community.

- In Chapter 4, we proposed the essential management tools to realize the promising properties of SoftAir. Specifically, first, given single-controller SoftAir, the control traffic balancing was proposed that enabled timely delivery of in-band control messages with minimum average delay. The simulation results confirmed that our design successfully demonstrated communication efficiency with at least 80% delay reduction via a fast and low complexity approach. Second, regarding multiple-controllers SoftAir, the optimal network planning was proposed that jointly optimized control traffic balancing and controller placement with minimum required controllers and the control traffic delay. The results showed that the proposed control scheme, based on the minimum number of required controllers, demonstrated communication efficiency with at least 73% delay reduction, close to the benchmark performance. Third, the network virtualization was introduced that provided a multi-tenancy management framework to enable the jointly optimized design of QoS-aware virtualization and routing by tenant isolation and prioritization as well as flow allocation, fulfilling QoS requirements of tenants' applications. Performance evaluation showed that our design outperformed the conventional approaches with less shared edges, congestion latency, and traffic delay for multi-tenants. Forth, the traffic classifier was proposed that jointly exploited deep packet inspection and semi-supervised machine learning so that accurate traffic classification can be realized, while requiring minimal communication between the network controller and SD-switches (SD-BSs). Based on the real Internet data set, the simulation results show the proposed classification framework can provide good performance in terms of classification accuracy and communication costs.

- In Chapter 5, we developed several novel software-defined traffic engineering solutions for SoftAir. In particular, first, through W-SDN SoftAir architecture, a dynamic BS formation was developed that achieved a joint optimal design of RRH-UE associations and beamforming weights of millimeter-wave RRHs to facilitate ubiquitous millimeter-wave coverage. Simulations showed that our solution can always satisfied UEs' QoS requirements with millimeter-wave transmissions and significantly outperformed conventional association schemes with suboptimal beamforming. Second, a throughput-optimal and delay-based maximum power-weight scheduling for single-hop flows with heavy-tailed traffic was introduced. Specifically, the DMPWS and its variant IU-DMPWS were proposed and proved to be of throughput optimality with respect to moment stability. The IU-DMPWS policy was also favored among scheduling algorithms for practical implementation because of its immunity to the impact of HT traffic along with limited signaling overhead. Third, LIFO-DMWS was proposed to achieve throughput-optimality with heavy-tailed traffic. In particular, LIFO-DMWS guaranteed bounded average delay for hybrid HT and LT flows under any admissible traffic arrivals without any knowledge of statistical information of the arrivals. Forth, QAR algorithm was proposed as network service via RL in multi-layer hierarchical SoftAir. Performance evaluation confirmed that QAR outperformed the conventional Q-learning approach with fast sublinear convergence when considering QoS provisioning.

Through the synergy of SDN and NFV solutions, the developed SoftAir in this thesis lays out the foundation for 5G wireless software-defined cellular systems.

PUBLICATIONS

Journal Papers

- S.-C. Lin, P. Wang, and M. Luo, "Control Traffic Balancing in Software Defined Networks," *Computer Networks*, vol. 106, pp. 260-271, Sept. 2016.
- I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, "5G Roadmap: 10 Key Enabling Technologies," *Computer Networks*, vol. 106, pp. 17-48, Sept. 2016.
- S.-C. Lin, P. Wang, and M. Luo, "Jointly Optimized QoS-Aware Virtualization and Routing in Software Defined Networks, *Computer Networks*, vol. 96, pp. 69-78, Feb. 2016.
- I. F. Akyildiz, S.-C. Lin, and P. Wang, "Wireless Software-Defined Networks (W-SDNs) and Network Function Virtualization (NFV) for 5G Cellular Systems: An Overview and Qualitative Evaluation," *Computer Networks*, vol. 93, pp. 66-79, Dec. 2015.
- I. F. Akyildiz, P. Wang, and S.-C. Lin, "SoftAir: A Software Defined Networking Architecture for 5G Wireless Systems," *Computer Networks*, vol. 85, pp. 1-18, July 2015.
- S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Delay-based Maximum Power-weight Scheduling in Queueing Networks with Heavy-tailed Traffic," revised for publication in *IEEE/ACM Trans. Netw.*, Feb. 2017. [Online]. Available: `http://www.prism.gatech.edu/~slin88/publications/DMPWS.pdf`.
- S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Towards Optimal Network Planning for Software-defined Networks," revised for publication in *IEEE Trans. Mobile Comput.*, Feb. 2017. [Online]. Available: `http://www.prism.gatech.edu/~slin88/publications/ONP.pdf`.

Conference Papers

- A. Gran, S.-C. Lin, and I. F. Akyildiz, "Towards Wireless Infrastructure-as-a-Service (WIaaS) for 5G Software-defined Cellular Systems," accepted by *Proc. of IEEE ICC17*, Paris, France, May 2017.
- S.-C. Lin and I. F. Akyildiz, "Dynamic Base Station Formation for Solving NLOS Problem in 5G Millimeter-wave Communication," accepted by *Proc. of IEEE INFOCOM17*, Atlanta, Georgia, USA, May 2017.
- S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Throughput-optimal LIFO Policy for Bounded Delay in the Presence of Heavy-tailed Traffic," in *Proc. IEEE GLOBECOM16*, Dec. 2016. [Online].
- S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "QoS-aware Adaptive Routing in Multi-layer Hierarchical Software Defined Networks: A Reinforcement Learning Approach," in *Proc. IEEE SCC16*, San Francisco, USA, June 2016.
- P. Wang, S.-C. Lin, and M. Luo, "A Framework for QoS-aware Traffic Classification Using Semi-supervised Machine Learning in SDNs," in *Proc. IEEE SCC16*, San Francisco, USA, June 2016.
- A. X. Porxas, S.-C. Lin, and M. Luo, "QoS-aware Virtualization-enabled Routing in Software Defined Networks, in *Proc. IEEE ICC15*, London, UK, June 2015.

REFERENCES

[1] L. Li, Z. Mao, and J. Rexford, "Toward software-defined cellular networks," in *2012 European Workshop on Software Defined Networking (EWSDN)*, pp. 7–12, Oct 2012.

[2] I. F. Akyildiz, P. Wang, and S.-C. Lin, "SoftAir: A software defined networking architecture for 5G wireless systems," *Computer Networks*, vol. 85, pp. 1–18, July 2015.

[3] I. F. Akyildiz, S.-C. Lin, and P. Wang, "Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: An overview and qualitative evaluation," *Computer Networks*, vol. 93, Part 1, pp. 66–79, 2015.

[4] Internet2, "Open science, scholarship and services exchange," 2013. [Online]. Available: http://www.internet2.edu/network/ose.

[5] Sprint, Overland Park, KS, "Sprint IP network performance," 2011. [Online]. Available: http://www.sprint.net/performance.

[6] ICT-317669 METIS project, "Simulation guidelines," Nov. 2013. Deliverable D6.1.

[7] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, pp. 1539–1552, Oct. 2013.

[8] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press, 1988.

[9] E. Hossain and M. Hasan, "5G cellular: key enabling technologies and research challenges," *IEEE Instrumentation Measurement Magazine*, vol. 18, pp. 11–21, June 2015.

[10] G. T. 25.913, "Requirements for Evolved UTRA and Evolved UTRAN," December 2009.

[11] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?," *IEEE J. Sel. Areas Commun.*, vol. 32, pp. 1065–1082, June 2014.

[12] G. Fettweis, "The Tactile Internet: Applications and Challenges," *IEEE Vehicular Technology Magazine*, vol. 9, pp. 64–70, March 2014.

[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, pp. 2347–2376, Fourthquarter 2015.

[14] "Cisco visual networking index: Forecast and methodology, 2014–2019," *CISCO White paper*, May 2015.

[15] R. Di Taranto, S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson, and H. Wymeersch, "Location-Aware Communications for 5G Networks: How location information can improve scalability, latency, and robustness of 5G," *IEEE Signal Processing Magazine*, vol. 31, pp. 102–112, Nov 2014.

[16] X. Duan and X. Wang, "Authentication handover and privacy protection in 5G hetnets using software-defined networking," *IEEE Communications Magazine*, vol. 53, pp. 28–35, April 2015.

[17] A. Ruiz-Martinez, "Towards a web payment framework: State-of-the-art and challenges," *Electronic Commerce Research and Applications*, vol. 14, no. 5, pp. 345 – 350, 2015. Contemporary Research on Payments and Cards in the Global Fintech Revolution.

[18] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Netw.*, vol. 30, pp. 52–58, May 2016.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Apr. 2008.

[20] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 236–262, First-quarter 2016.

[21] "ETSI GS NFV 003 V1.2.1: Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV." ETSI Industry Specification Group (ISG) NFV, Dec. 2014.

[22] "Common public radio interface (CPRI) specification v6.0; interface specification," Aug. 2013.

[23] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," in *Proceedings of the ACM SIGCOMM 2010 conference*, (New York, NY, USA), pp. 351–362, ACM, 2010.

[24] M. Casado, M. J. Freedman, J. P. J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proceedings of the ACM SIGCOMM 2007 conference*, (New York, NY, USA), pp. 1–12, 2007.

[25] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for introducing innovation into wireless mobile networks," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pp. 25–32, ACM, 2010.

[26] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: a programmable wireless dataplane," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 109–114, ACM, 2012.

[27] P. Dely, J. Vestin, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud," in *2012 IEEE Globecom Workshops (GC Wkshps)*, pp. 186–191, Dec 2012.

[28] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 115–120, ACM, 2012.

[29] "SDN/NFV cloud computing platform and core network for 5G services," tech. rep., Centre Tecnologic de Telecomunicacions de Catalunya (CTTC).

[30] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 25–30, ACM, 2013.

[31] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Network*, vol. 29, pp. 35–41, Jan 2015.

[32] NTT DOCOMO, INC, "DOCOMO 5G white paper, 5G radio access: Requirements, concept and technologies," tech. rep., July 2014.

[33] SK Telecom, "SK Telecom 5G white paper, SK Telecom's view on 5G vision, architecture, technology, and spectrum," tech. rep., 2014.

[34] "Project CONTENT FP," 2012-2015.

[35] S.-C. Lin, P. Wang, and M. Luo, "Control traffic balancing in software defined networks," *Computer Networks*, vol. 106, pp. 260–271, 2016.

[36] ONF, "OpenFlow switch specification," version 1.4.0.

[37] C. Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *ACM SIGCOMM*, pp. 15–26, Aug. 2013.

[38] G. Bolch, S. Greiner, H. Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, Second Edition*. New York: Wiley, 2006.

[39] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*. Wiley-Sons, 3rd ed., 2008.

[40] D. S. Hochbaum, "Complexity and algorithms for nonlinear optimization problems," *Annals OR*, vol. 153, no. 1, pp. 257–296, 2007.

[41] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[42] S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Towards optimal network planning for software-defined networks," submitted for publication in IEEE Trans. Mobile Comput., Oct. 2015. [Online]. Available: http://www.prism.gatech.edu/~slin88/publications/ONP.pdf.

[43] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 2665–2670, Dec. 2012.

[44] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, 2004.

[45] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms, second edition*. MIT Press, 2001.

[46] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *ACM HotSDN*, Aug. 2012.

[47] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, pp. 38–54, Feb. 2014.

[48] S. Lange, S. Gebert, T. Zinner, P. T.-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management*, vol. 12, pp. 4–17, Mar. 2015.

[49] ANSI, "Enhanced network survivability performance," Tech. Rep. T1.TR.68-2001.

[50] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, Oct. 2014.

[51] D. Shue, M. J. Freedman, and A. Shaikh, "Performance isolation and fairness for multi-tenant cloud storage," in *10th USENIX Conf. Operat. Syst. Design Implement*, p. 349V362, Oct. 2012.

[52] W.-T. Tsai, Q. Shao, and J. Elston, "Prioritizing service requests on cloud with multitenancy," in *IEEE 7th International Conference on e-Business Engineering (ICEBE)*, pp. 117–124, Nov. 2010.

[53] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *2013 Proceedings IEEE INFOCOM*, pp. 2211–2219, Apr. 2013.

[54] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *ACM SIGCOMM*, pp. 3–14, Aug. 2013.

[55] M. Garey and D. Johnson, *M. Garey and D. Johnson*. Freeman, 1979.

[56] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, pp. 756–767, May 2002.

[57] J. Chu and C.-T. Lea, "Optimal link weights for ip-based networks supporting hose-model vpns," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 778–788, 2009.

[58] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," in *the American Mathematical Society*, 1956.

[59] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, pp. 269–271, 1959.

[60] R. Sherwood, G. Gibby, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," Tech. Rep. OPENFLOW-TR-2009-1, Deutsche Telekom Inc. R&D Lab., Stanford University, Nicira Networks, 2009.

[61] J. Moy, "Ospf version 2," *IETF RFC 2328*, 1998.

[62] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 1999.

[63] G. Camps-Valls, T. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45.10, pp. 3044–3054, 2007.

[64] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *The Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[65] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[66] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *The Journal of Machine Learning Research*, vol. 12, pp. 1149–1184, 2011.

[67] J. Erman and et al., "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64.9, pp. 1194–1213, 2007.

[68] I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, "5G roadmap: 10 key enabling technologies," *Computer Networks*, vol. 106, pp. 17–48, Sept. 2016.

[69] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5G cellular: It will work!," *IEEE Access*, vol. 1, pp. 335–349, 2013.

[70] M. R. Akdeniz, Y. Liu, M. K. Samimi, S. Sun, S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE J. Sel. Areas Commun.*, vol. 32, pp. 1164–1179, June 2014.

[71] T. Bai, A. Alkhateeb, and R. W. Heath, "Coverage and capacity of millimeter-wave cellular networks," *IEEE Commun. Mag.*, vol. 52, no. 9, pp. 70–77, 2014.

[72] Y. Cheng, M. Pesavento, and A. Philipp, "Joint network optimization and downlink beamforming for CoMP transmissions using mixed integer conic programming," *IEEE Trans. Signal Process.*, vol. 61, pp. 3972–3987, Aug. 2013.

[73] O. Tervo, L. N. Tran, and M. Juntti, "Decentralized coordinated beamforming for weighted sum energy efficiency maximization in multi-cell MISO downlink," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1387–1391, Dec. 2015.

[74] T. Bai and R. W. Heath, "Asymptotic sinr for millimeter wave massive MIMO cellular networks," in *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 620–624, June 2015.

[75] A. Beck, A. Ben-Tal, and L. Tetruashvili, "A sequential parametric convex approximation method with applications to nonconvex truss topology design problems," *Journal of Global Optimization*, vol. 47, no. 1, pp. 29–51, 2010.

[76] IBM ILOG CPLEX Optimizer Version l v12.4.

[77] MOSEK Optimization Software Version 7.0.

[78] M. Luo, S.-C. Lin, and I. F. Akyildiz, "Traffic-driven network controller placement in software-defined networks," Apr. 28 2016. US Patent App. 15/141,367.

[79] M. D. Renzo, "Stochastic geometry modeling and analysis of multi-tier millimeter wave cellular networks," *IEEE Trans. Wireless Commun.*, vol. 14, pp. 5038–5057, Sept. 2015.

[80] S.-C. Lin and I. F. Akyildiz, "Dynamic base station formation for solving nlos problem in 5g millimeter-wave communication," 2017. accepted by *Proc. of IEEE INFOCOM17*.

[81] J. Jose, A. Ashikhmin, T. L. Marzetta, and S. Vishwanath, "Pilot contamination and precoding in multi-cell TDD systems," *IEEE Trans. Wireless Commun.*, vol. 10, no. 8, pp. 2640–2651, 2011.

[82] V. Ha, L. Le, and N. D. Dao, "Coordinated multipoint (CoMP) transmission design for cloud-RANs with limited fronthaul capacity constraints," *IEEE Trans. Veh. Technol.*, 2015.

[83] P. Baracca, F. Boccardi, and V. Braun, "A dynamic joint clustering scheduling algorithm for downlink CoMP systems with limited CSI," in *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 830–834, Aug. 2012.

[84] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, pp. 1936–1948, Dec. 1992.

[85] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM Trans. Netw.*, vol. 21, pp. 41–54, Feb. 2013.

[86] P. van de Ven, S. Borst, and S. Shneer, "Instability of maxweight scheduling algorithms," in *2009 IEEE INFOCOM*, pp. 1701–1709, Apr. 2009.

[87] K. Kar, X. Luo, and S. Sarkar, "Throughput-optimal scheduling in multichannel access point networks under infrequent channel measurements," *IEEE Trans. Wireless Commun.*, vol. 7, pp. 2619–2629, July 2008.

[88] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 411–424, Apr. 2005.

[89] D. J. Daley, "The moment index of minima," *J. Appl. Probab.*, vol. 38A, pp. 33–36, 2001.

[90] S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Delay-based maximum power-weight scheduling in queueing networks with heavy-tailed traffic," submitted for publication in IEEE/ACM Trans. Netw., Mar. 2016. [Online]. Available: http://www.prism.gatech.edu/~slin88/publications/DMPWS.pdf.

[91] M. Bramson, "Stability of queueing networks," *Probab. Surv.*, vol. 5, pp. 169–345, 2008.

[92] P. Wang and I. F. Akyildiz, "On the stability of dynamic spectrum access networks in the presence of heavy tails," *IEEE Trans. Wireless Commun.*, vol. 14, no. 2, pp. 870–881, 2015.

[93] S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Throughput-optimal LIFO policy for bounded delay in the presence of heavy-tailed traffic," to appear in Proc. IEEE GLOBECOM16, Dec. 2016. [Online]. Available: http://www.prism.gatech.edu/~slin88/publications/LIFODMWS.pdf.

[94] P. Wang and I. F. Akyildiz, "Asymptotic queueing analysis for dynamic spectrum access networks in the presence of heavy tails," *IEEE J. Sel. Areas Commun.*, vol. 3, no. 3, pp. 514–522, 2013.

[95] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *OSDI*, 2010.

[96] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *ACM HotSDN*, pp. 19–24, Aug. 2012.

[97] J. McCauley, A. Panda, M. Casado, T. Koponen, and S. Shenker, "Extending SDN to large-scale networks," in *ONS*, Mar. 2013.

[98] M. Luo, Q. Li, M. Bo, K. Lin, X. Wu, C. Li, S. Lu, and W. Chou, "Design and implementation of a scalable SDN-OF controller cluster," in *INFOCOMP*, June 2015.

[99] S. C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *2016 IEEE International Conference on Services Computing (SCC)*, pp. 25–33, June 2016.

[100] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," tech. rep., 1994.

[101] M. N. Katehakis and A. F. Veinott, *The Multi-Armed Bandit Problem: Decomposition and Computation*. Mathematics of OR, 1987.

[102] H. A. Tran, A. Mellouk, S. Hoceini, and B. Augustin, "Global state-dependent QoE based routing," in *2012 IEEE International Conference on Communications (ICC)*, pp. 131–135, June 2012.

[103] B. F. Lo and I. F. Akyildiz, "Reinforcement learning for cooperative sensing gain in cognitive radio ad hoc networks," *Wireless Networks, Springer*, vol. 19, no. 6, pp. 1237–1250, 2013.

[104] M. Luo, Y. Zeng, J. Li, and W. Chou, "An adaptive multi-path computation framework for centrally controlled networks," *Computer Networks*, vol. 83, pp. 30–44, 2015.

VITA

Shih-Chun Lin received the Bachelor of Science degree in Electrical Engineering and the Master of Science degree in Communication Engineering from National Taiwan University, Taipei, Taiwan, in 2008 and 2010, respectively. He received the Ph.D. degree in School of Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, USA, in May 2017, under the supervision of Prof. Ian F. Akyildiz. He is a member of the IEEE. His current research interests include wireless software-defined networking, network function virtualization, wireless communication in challenged underground and underwater environments, and Internet of Things, focusing on statistical scheduling, MAC- and network-layer designs, and mathematical optimization.